DEVELOPMENT OF VIRTUAL MITRAL VALVE LEAFLET MODELS FROM THREE-DIMENSIONAL ECHOCARDIOGRAPHY

A Thesis
Presented to
The Academic Faculty

by

David Andrew Icenogle

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Bioengineering

Georgia Institute of Technology
August, 2012

DEVELOPMENT OF VIRTUAL MITRAL VALVE LEAFLET MODELS FROM THREE-DIMENSIONAL ECHOCARDIOGRAPHY

Approved by:

Dr. Ajit Yoganathan, Advisor
School of Biomedical Engineering
Georgia Institute of Technology

Dr. Jarek Rossignac
College of Computing
Georgia Institute of Technology

Dr. Robert Guldberg
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: 6/29/12

For my wife, Cassilyn.

ACKNOWLEDGEMENTS

It has been a long journey, but it is finally done. There have been many that have helped me along the way. First, I would like to thank Dr. Ajit Yoganathan for taking me on as a graduate student and pushing me to develop into a better engineer. Second, I would like to thank Dr. Jarek Rossignac, who showed me the wonderful world of computer graphics and algorithms. While I may not continue professionally in that realm, I will always have a passion for it. I would also like to acknowledge the final member of my committee, Dr. Guldberg.

In addition, Holifield Farms in Covington, GA. Without their donation of porcine hearts to our lab this and many other works would not be possible. I must express my thanks and gratitude to all the members of the Cardiovascular Fluid Mechanics Lab who have helped along the way. Murali, Prasad, Neela, and JP, without your help I would not have been able to complete this work and I greatly appreciate it. All the other members of the lab who all helped in one way or another: giving feedback, ideas, laughs, or just good conversation. Andrew, Yap, Chris, Kartik B., Kartik S., Brandon, Jorge, Shiva, Helene, Diane, and any of those I may have forgotten at this time, I thank you for the opportunity to learn and laugh with you. It has been a pleasure.

Finally, and most importantly, I would like to acknowledge my wife for her unending love and support through the good times and bad. Cass, I dedicate this work to you and our family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Worldwide, cardiovascular disease is the leading cause of death accounting for over 16 million deaths in 2004, which made up over 27% of all deaths[1]. Mitral valve disease is a subset of cardiovascular disease. In the United States, there are approximately 2,581 deaths and 41,000 hospital discharges related directly to mitral valve disorders each year[2]. Moderate mitral regurgitation (MR) occurred in at least 1.7% of the US adult population in 2000[2]. At age 18, 0.5% have moderate MR, while 9.3% over the age of 75 have moderate MR[2]. In addition, about 2.4% of the US population has mitral valve prolapse[2].

In current cardiothoracic surgical practice, mitral valve repair is preferred over replacement as it leads to improved quality of life and does not require lifelong blood thinners. Since the 1980s, the percentage of mitral valve repairs has increased from 5% to 59.78% in 2008[3]. While repair has become the standard of care for mitral valve disease, the long term patient outcomes have resulted in 15% to 80% of patients having recurrent mitral regurgitation within 10 years of operation[3]. The failure mechanism for these repairs is currently unknown.

In parallel, with a shift in the standard of care to mitral valve repair, real-time 3D echocardiography (RT3DE) has made advances to generate high quality images of the mitral valve with high spatial and temporal resolution. While RT3DE images are used to visualize the mitral valve in 3D, it is not used to quantify the 3D geometry of the mitral leaflets to better understand disease mechanisms. Quantification of 3D mitral leaflet geometry could be accomplished if virtual models of the leaflets could be generated from the RT3DE images.

The overall objective of the present study was to create virtual geometric models of the dynamic mitral valve leaflets and annulus from RT3DE images. A manual segmentation tool was developed that was capable of segmenting the mitral valve from RT3DE images throughout the cardiac cycle. Once the tool was fully developed, the method was validated using an *in vitro* mitral valve model under normal, flail, and billowing leaflet conditions. This work is the first step toward patient specific modeling of the dynamic mitral valve, with the long term objective of providing patient specific inputs to finite element analysis (FEA) and computation fluid dynamics (CFD) models to optimize mitral repair procedures.

CHAPTER 2

BACKGROUND

2.1 The Heart

The heart is the organ responsible for moving blood throughout the body. It is comprised of four chambers and four valves. Each chamber pumps the blood, while the valves ensure unidirectional flow of the blood. The heart can be separated into low and high pressure systems. The low pressure system is the right side of the heart and is comprised of the right atrium, right ventricle, tricuspid valve, and the pulmonary valve. The right side of the heart is responsible for moving deoxygenated blood from the venous circulation to the lungs to be oxygenated. The high pressure system is the left side of the heart and is composed of the left atrium, left ventricle, mitral valve, and aortic valve. The left side of the heart is responsible for moving oxygenated blood from the lungs to the systemic circulation (Figure 2.1).

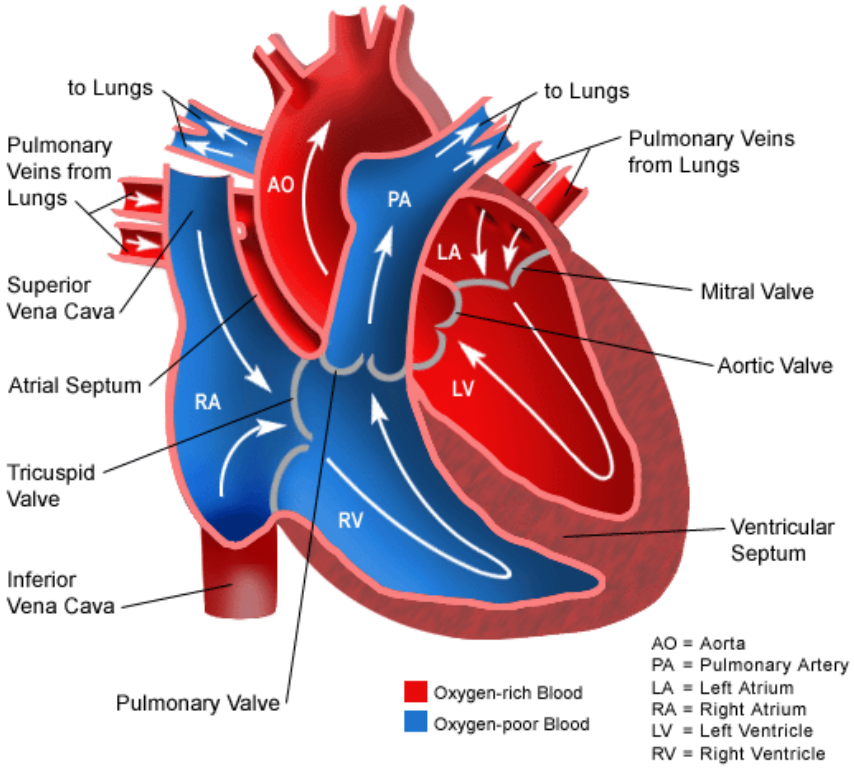


Figure 2.1 Diagram of the components of the heart and how blood flows through (http://www.schneiderchildrenshospital.org/peds_html_fixed/images/ei_0329.gif).

The right atrium and right ventricle are separated by the tricuspid valve. This valve is comprised of an annulus, three leaflets, chordae tendineae, and three papillary muscles. These structures are dynamic and function in unison to create proper closing and opening of the valve. During systole, the tricuspid valve is closed and prevents backflow from the right ventricle into the right atrium. During diastole, the tricuspid valve is open and allows blood to flow from the right atrium into the right ventricle. The pulmonary valve consists of three semilunar cusps. This valve controls blood flow between the right ventricle and the lungs. During systole, the pulmonary valve is open and allows blood to flow from the right ventricle to the lungs. During diastole, the valve is closed and prevents backflow of blood from the lungs into the right ventricle.

The left atrium and the left ventricle are separated by the mitral valve. This valve is comprised of an annulus, two leaflets, chordae tendineae, and two papillary muscles. These structures are dynamic and function in unison to create proper closing and opening of the valve. During systole, the mitral valve is closed and prevents backflow from the left ventricle to the left atrium. During diastole, the mitral valve is open and allows blood to flow from the left atrium to the left ventricle. The aortic valve consists of three semilunar cusps. This valve controls blood flow between the left ventricle and the systemic circulation. During systole, the aortic valve is open and allows blood to flow from the left ventricle to the systemic circulation. During diastole, the aortic valve is closed to prevent backflow from the systemic circulation into the left ventricle.

The right and left atria assist in moving blood from the systemic and pulmonary circulations into their respective ventricles. The two atria are similar in shape and structure, while the left and right ventricles have different shapes and structure. The right ventricle generates pressures up to 40 mmHg, has relatively thin walls, and has a crescent shaped cross section. The left ventricle generates pressures up to 120 mmHg, has relatively thick walls compared to the right ventricle, and has a circular shaped cross section.

2.2 The Mitral Valve

The mitral valve consists of an annulus, two leaflets, chordae tedineae, and two papillary muscles. The valve maintains unidirectional flow from the left atrium to the left ventricle. The annulus connects the leaflets to the left ventricle myocardium, while the chordae tendineae connect the leaflets to the papillary muscles. The mitral valve structures are dynamic and work in unison to ensure proper valve opening and closure throughout the cardiac cycle.

2.2.1 Mitral Valve Leaflets

The mitral valve is comprised of a continuous veil of tissue around the mitral orifice that is connected to the annulus. The veil of tissue is subdivided into two leaflets and two commissural areas. The two leaflets are the anterior and posterior, while the two commissural areas are designated as the anterolateral and posteromedial (Figure 2.2). The anterior leaflet is the larger of the two leaflets and is defined between the commissures connected to the aortic root. The posterior leaflet is defined between the commissures along the free wall of the ventricle. The commissural areas are characterized by fan-like chordal insertions.

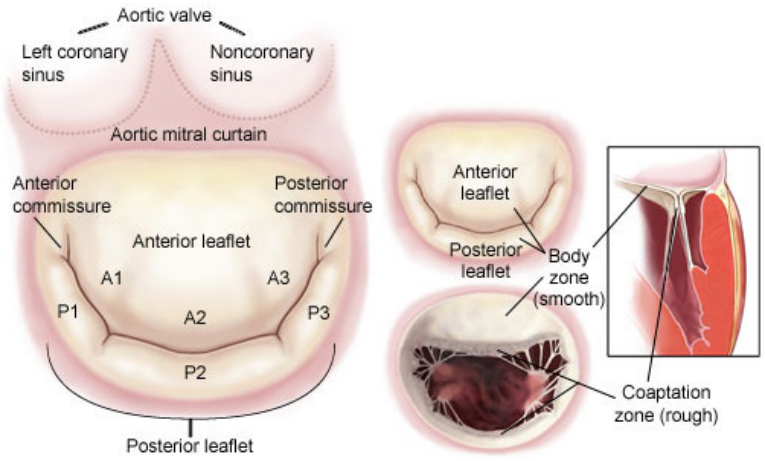


Figure 2.2 Diagram of the mitral valve leaflets and coaptation zone (http://www.mitralvalverepair.org/).

There are multiple zones of the anterior and posterior leaflets as shown in Figure 2.2. These consist of the basal, body, and rough zones. The rough zone is defined from the line of coaptation to the free edge of each leaflet.

2.2.2 Mitral Annulus

The mitral annulus is the connective interface between the mitral leaflets and the left ventricle. It is a dynamic structure that changes shape and size during the cardiac

cycle. During systole, the annulus contracts and forms a saddle shape[4] (Figure 2.3); during diastole, it dilates and adopts a flattened ring shape. These changes aid in closure of the valve during systole and facilitates ventricular filling during diastole.

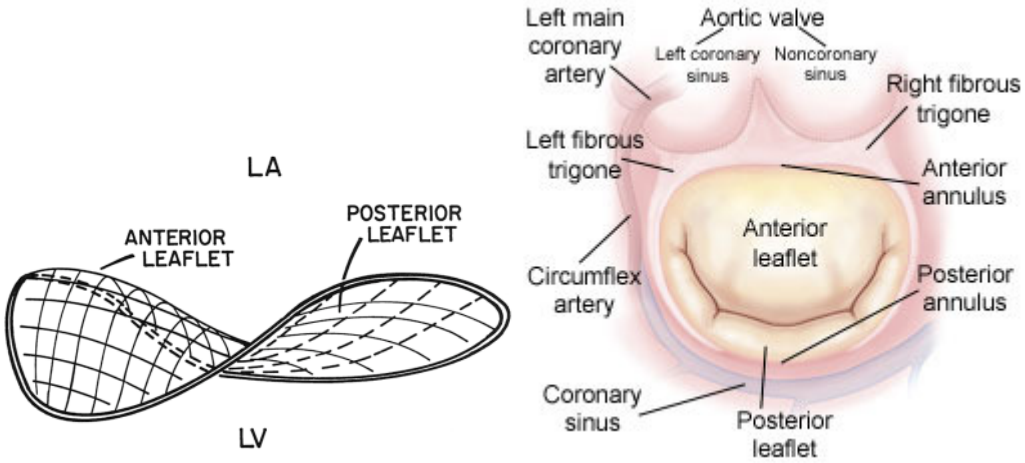


Figure 2.3 Saddle shape of the mitral annulus during peak systole (http://www.mitralvalverepair.org/)[4].

2.2.3 Chordae Tendineae

The chordae tendineae connect the valve leaflets to the papillary muscles and help ensure proper valve closure and prevent leaflet prolapse. They are normally classified into three different categories based upon their insertion points (Figure 2.4):

1) Primary or marginal, which insert into the free edge of the leaflets
2) Secondary, strut, or intermediate, which insert into the body of the leaflets
3) Tertiary, basal, or commissural, which insert into the basal region of the leaflets

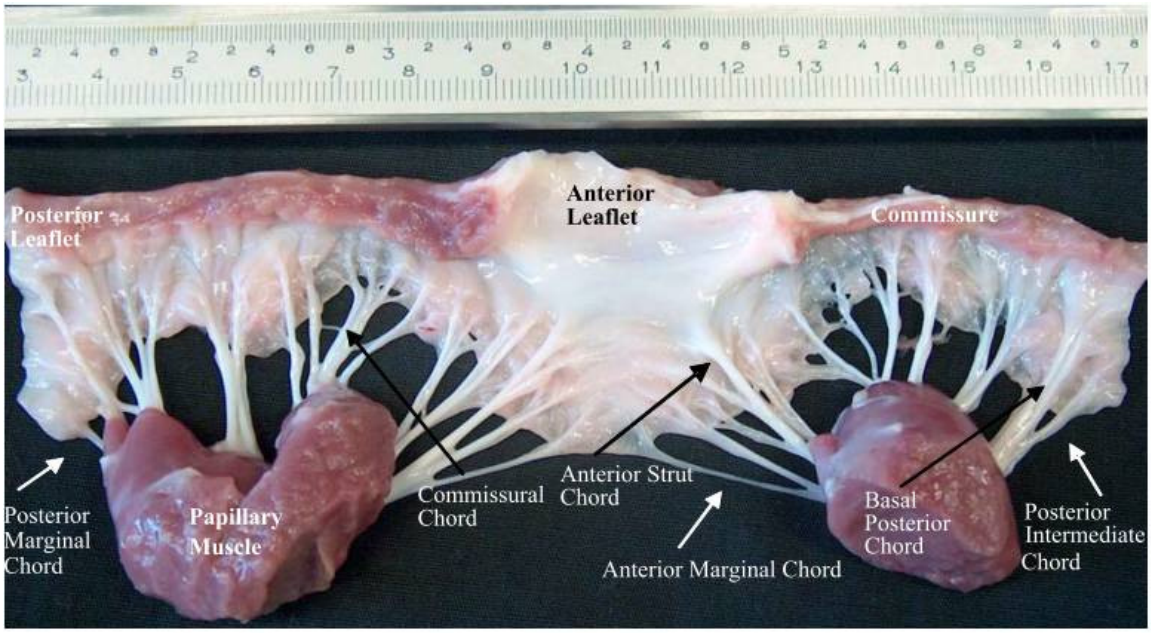


Figure 2.4 Diagram of the chordae tendineae of a porcine mitral valve.

In addition to having different insertion regions, each type of chord has varying lengths and thicknesses. The length and thickness of each type of chord is displayed in Table 2.1.

Table 2.1 Length and thickness of chordae tendineae based upon insertion site[5]

Site of Insertion	Types of Chordae	Length (cm)	Thickness (mm)
Anterior leaflet	Rough zone Strut	1.75 ± 0.25 1.86 ± 0.43	0.84 ± 0.28 1.24 ± 0.51
Posterior leaflet	Rough zone Basal Cleft	1.40 ± 0.08 0.84 ± 0.21 1.30 ± 0.18	0.65 ± 0.24 0.40 ± 0.29 0.78 ± 0.15
Commissural areas	Anterior lateral Posterior medial	1.20 ± 0.31 1.40 ± 0.40	0.70 ± 0.20 1.00 ± 0.30

2.2.4 Papillary Muscles

The papillary muscles connect the chordae tendineae to the left ventricle as shown in Figure 2.5. There are two sets of papillary muscles: the anterolateral and the

posteromedial. The chordae tendineae normally extend from the tips of the papillary muscles and insert into the leaflets. During systole, the papillary muscles contract in order to maintain a constant distance between the papillary muscles tips and the mitral annulus.

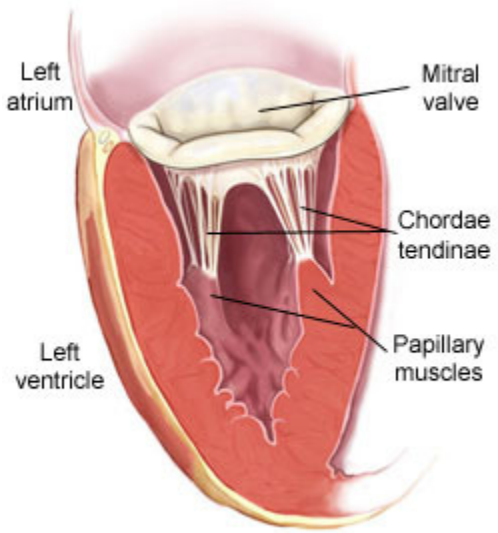


Figure 2.5 Diagram of the chordae tendineae and papillary muscles of the mitral valve. The anterolateral papillary muscle is on the left of the image and the posteromedial is on the right (http://www.mitralvalverepair.org/).

2.3 Echocardiography

2.3.1 Principles of Echocardiography

Echocardiography uses ultrasonic sound waves to image structures within the heart. The ultrasonic pulses are generated by piezoelectric crystals within the echo probe that vibrate at frequencies between 1 to 7 Mhz[6]. From the probe, an acoustic wave is generated that is partially reflected and transmitted at the boundary of two different tissue structures. The amount of reflection is dependent on the difference in the acoustic impedance of the two tissues[7]. Given the speed of sound in soft tissue is known to be 1540 m/s[8], the time of travel of a sound wave can be used to determine the distance

traveled. With the direction and distance of the sound beam known, the spatial location of the tissue can be determined.

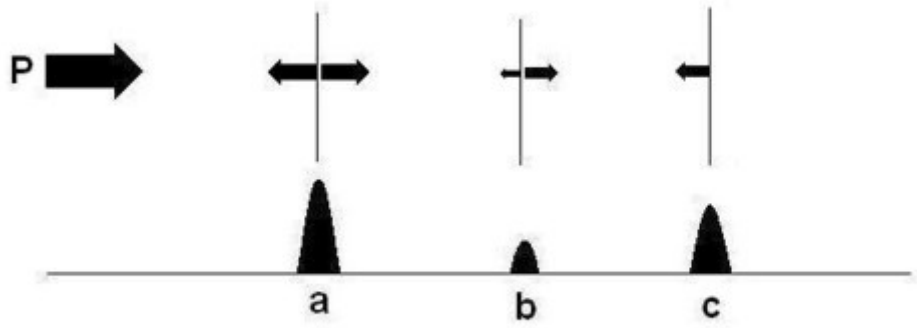


Figure 2.6 Schematic of an ultrasound pulse generated by a probe (P) and reflected at a, b, and c. The amplitude of the signal reflected at a is the largest, b the smallest, and c the intermediate[7].

The amplitude of the signal returned to the probe will determine the brightness of the area displayed in the echo image. This explains how an ultrasound measurement is taken for a single beam. The amplitude of the signal is also affected by attenuation, which can reduce the distinction between structures.

Attenuation is dependent upon two main factors: the depth and frequency of the ultrasound beam[6, 7]. The amplitude of the ultrasound signal sent will be reduced as depth increases. The half-power distances, the distance at which the amplitude of the signal is decreased by half, for blood and soft tissue are 15 cm and 5 cm respectively[6]. The higher the frequency of an ultrasound beam the greater the attenuation. In soft human tissue, the attenuation is generally approximated at 1 dB/cmMHz[7]. However, in imaging, a beam travels to and from the measured tissue. Thus, imaging at a depth of 5 cm will result in 10 cm beam travel distance. In general, higher frequencies are used for closer measurements. This is because the attenuation is low enough to use higher frequencies and higher frequencies result in better resolution. Typically, for adult

transthoracic echocardiography (TTE), a frequency of about 3.5 MHz is used, while for pediatric TTE, a frequency of about 5 MHz is used[7]. For transesophageal echocardiography (TEE), when the probe is much closer to the heart, a frequency of 5 to7 MHz would be optimal[6].

While the process of measuring with a single ultrasound beam has been discussed, many beams are sent out at prescribed angles and measured in succession when echocardiography measurements are taken. A phased array of transducers is used to change the direction of the ultrasound pulse sent out by the probe. A phased array of transducers is capable of creating a beam in multiple directions. This is accomplished by activating each transducer in the array at a specific time delay to create a wavefront in the desired direction according to Huygens principle (Figure 2.7)[7]. In addition to this beamforming, beam focusing is used to increase the concentration of the sound energy at the desired depth.

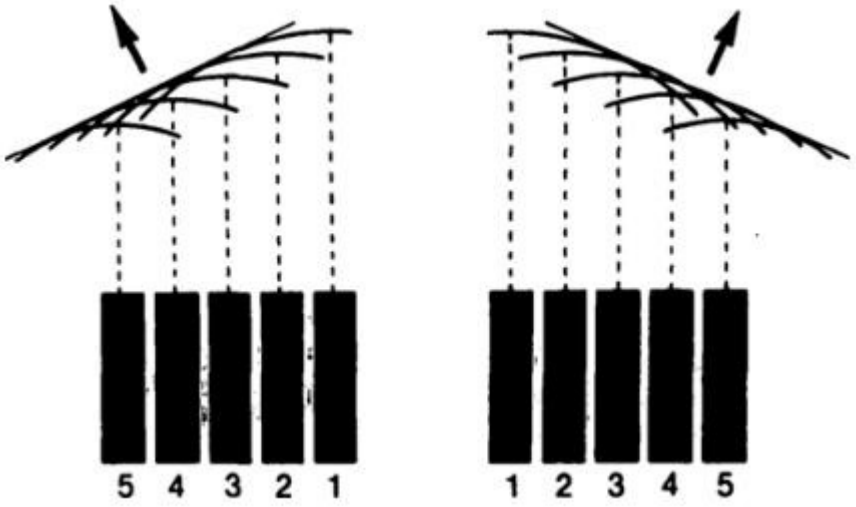


Figure 2.7 Phased linear array of transducers generating a wavefront by activating each transducer with a specific time delays to create the wavefront direction desired[6].

Beam focusing concentrates the sound energy of the ultrasound beam at a specific depth. This is accomplished by using a phased array to create a concave wavefront with

focuses the beam as shown in Figure 2.8. The focus occurs at the point where the wavefront is narrowest[7]. In practice, this allows structures at a particular depth to have greater reflection amplitudes than would be possible without the focused beam. Figure 2.9 depicts an example of the sound field for an unfocused and focused 4 MHz transducer in water.

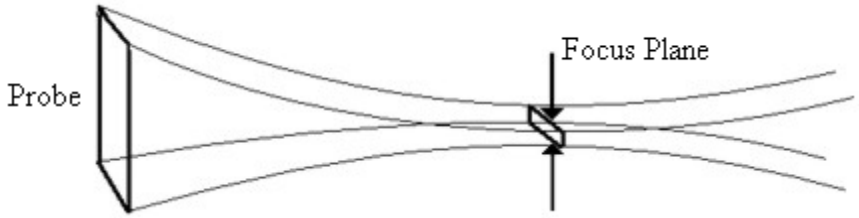


Figure 2.8 Example of the focus plane of a focused ultrasound beam [7].

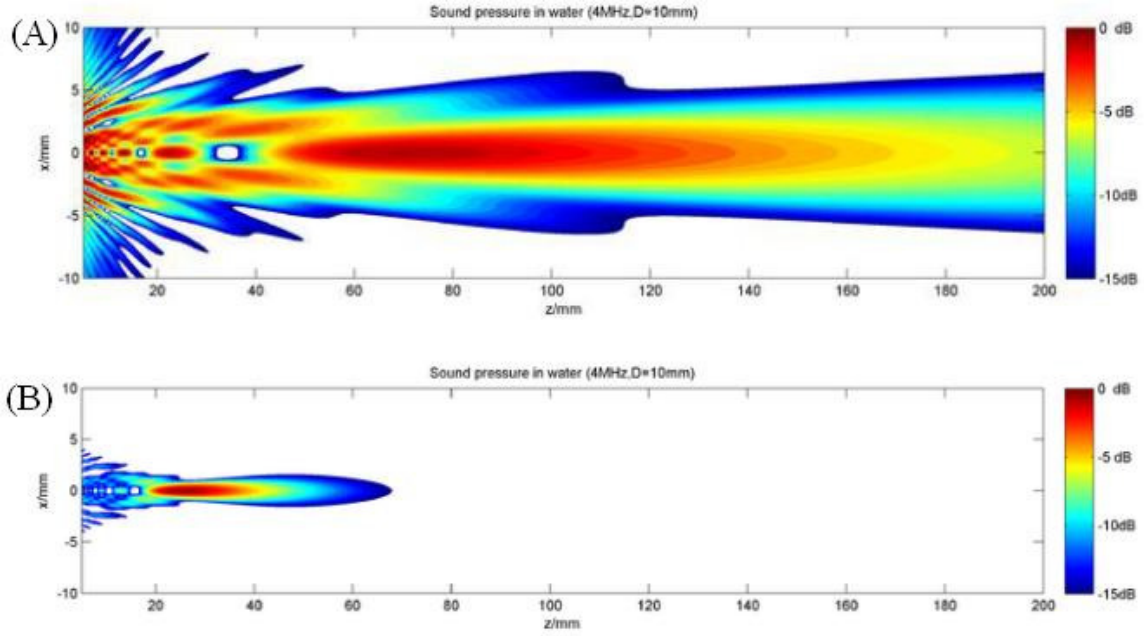


Figure 2.9 Example of the sound field from an unfocused (A) and focused (B) 4 MHz ultrasonic transducer[9].

Extending Huygens principle to a matrix array of transducers instead of a linear array (Figure 2.7) allows for the ultrasound beam to be formed in three dimensions

(Figure 2.10, Figure 2.11). Here the beam location is determined in spherical coordinates using two angles and the beam radius. The two angles are generated by the probe based upon Huygens principle and the radius is determined by the time of travel of the ultrasound beam.

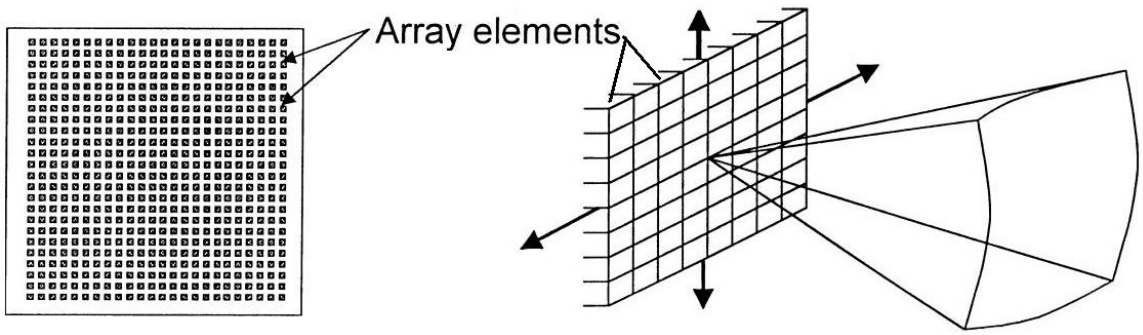


Figure 2.10 Schematic of a matrix array using to generate ultrasound beams for 3D echocardiography measurements[10].

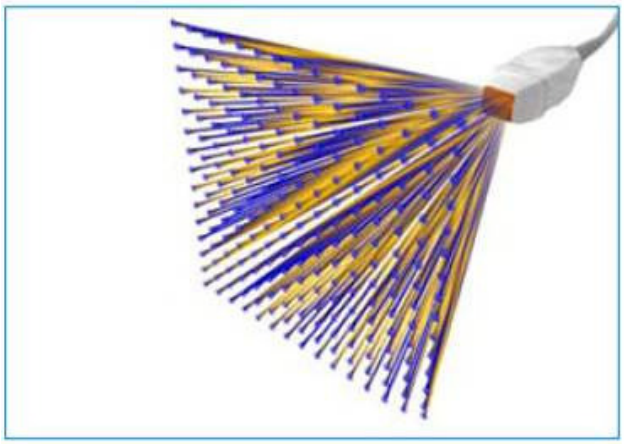


Figure 2.11 3D ultrasound field measured by a probe with a matrix array of transducers (http://www.healthcare.philips.com/in_en/products/ultrasound/technologies/xmatrix.wpd)

A key principle of echocardiography is that it can only measure what it on a single beam path at a time (Figure 2.6). Therefore, in order to survey a volume, a large number of ultrasound beams must be sent, received, and interpreted by the echocardiography machine. For example, if the ultrasound beams of an

echocardiography measurement are assumed to all have a constant speed of 1540 m/s, then you could have 1540 m of ultrasound beam travel for measurements gathered in a single second. Using a 10 cm measurement depth and converting, this would 154,000 cm of sound travel for measurement in one second. However, the beam has to travel 10 cm to the end of the measurement depth and 10 cm back to the probe for a total of 20 cm of beam travel to get a full measurement along that beam path. So, in one second only 7700 beam paths can be measured at a depth of 10 cm. This is the physical limit of ultrasound measurements at a depth of 10 cm in a medium with an assumed speed of sound of 1540 m/s.

The physical limitation of the number of measurements that ultrasound can take within a certain time can limit the resolution of tissue measurements when velocity measurements are for the same echocardiography image. It also limits the temporal resolution of "live" 3D ultrasound images compared to "live" 2D ultrasound images. In addition, there is significantly more data to process within the same time period for 3D ultrasound measurement. This may reach the limits of current software and hardware. There may also be some signal processing limitations as well. Signal processing is also a challenge because ultrasound forms a wavefront and not a distinct beam. This causes a significant amount of signal noise from reflections of the beam wavefront reflecting off of objects that are not in the "beam path".

While the physical limit will always be present in ultrasound measurements, improvements to software and hardware will continue yield ever improving imaging for clinical application. Better algorithms will allow for better filtering of raw ultrasound signals. Faster hardware will allow these algorithms to run more quickly, which could allow for less noise in the measurements and improved image quality.

2.3.2 Applications of Echocardiography

While 2D echocardiography remains in wide use, real-time 3D echocardiography (RT3DE) is rapidly becoming the standard for mitral valve imaging and diagnosis. In a 2D echocardiography exam, measurements of the ventricle and the mitral valve geometry are made from a single imaging plane. Measuring complex 3D anatomy with a single 2D plane can result in the structure of the heart being incorrectly characterized. A comparison of a 2D echo and 3D echo of a posterior flail leaflet is shown in Figure 2.12.

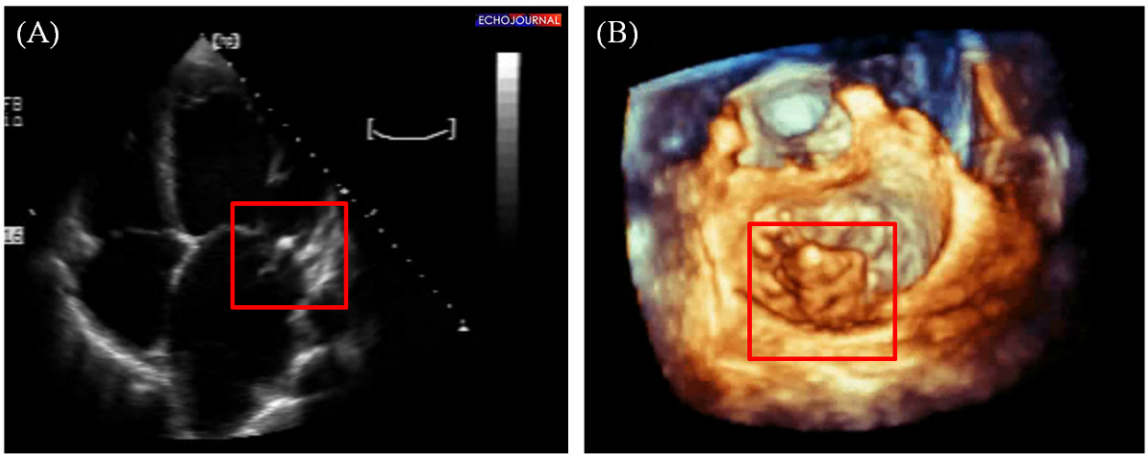


Figure 2.12 Example of 2D (A) and a RT3DE (B) of a posterior flail leaflet. (Image A – http://www.echojournal.org, Image B –http://www.healthcare.philips.com/).

Figure 2.12 provides a clear example of the advantages of RT3DE over standard 2D echocardiography. Since the entire valve can be imaged in 3D at one time, the severity of the flail leaflet can be better assessed prior to surgery. Even though RT3DE can provide complete 3D visualization of the mitral valve, 3D measures are not typically used to quantify the severity of disease or the underlying cause of the valve dysfunction. Some measures that are not possible with 2D echocardiography include leaflet area, leaflet curvature, and coaptation area.

In clinical practice 3D echocardiography has been shown to be an important tool for cardiologists and surgeons alike. In 2007, Garcia-Orta et al examined the use of 3D

echocardiography versus 2D transesophageal echocardiography in mitral valve repair. They found that 3D echocardiography was far superior at diagnosing mitral valve defects. It was capable of giving the surgeons a significant amount of additional information about the valve compared to 2D echocardiography. They noted this would particularly aid surgeons when more complex mitral valve repairs were undertaken. An example of the 3D echocardiography image compared with the surgical finding is shown in Figure 2.13[11].

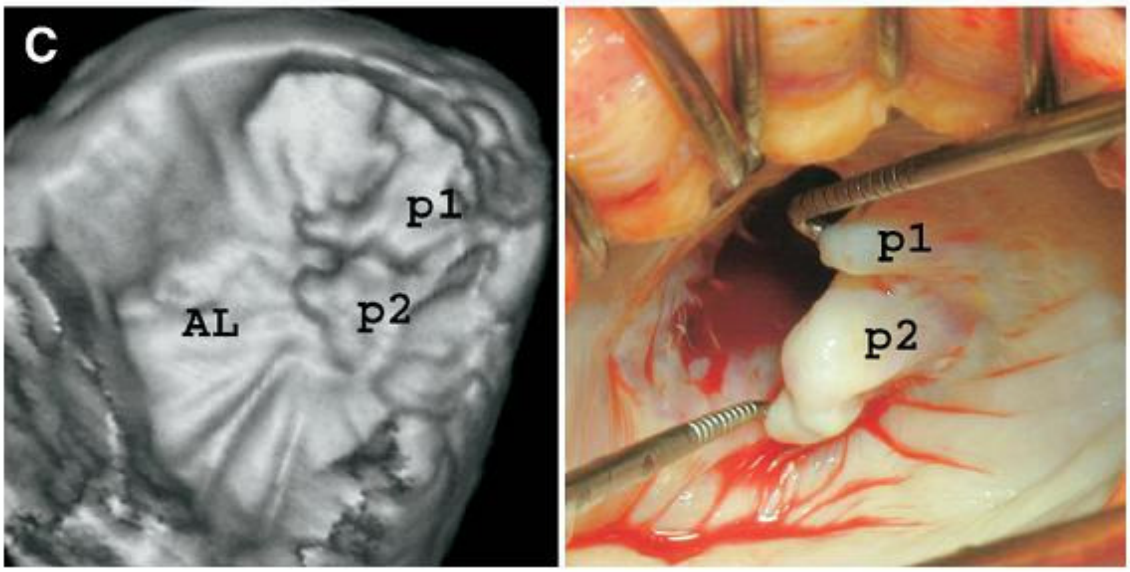


Figure 2.13 3D echocardiography image (left) compared with the associated surgical findings (right)[11].

Daimon et al evaluated the dynamic change in mitral annular area and motion in an animal study of percutaneous mitral annuloplasty with 3D echocardiography. 3D echocardiography allowed them to capture more accurate information about the valve. In fact, they would have been unable to dynamically capture the mitral area change over the cardiac cycle. They would have also been unable to calculate the mitral annular excursion (Figure 2.14)[12].

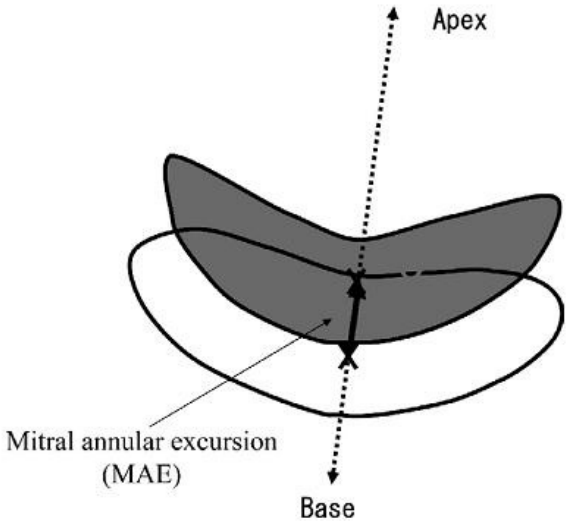


Figure 2.14 Mitral annular excursion (MAE), as defined by Daimon et al[12].

Their results (Figure 2.15) showed that the percutaneous treatment was able to return the mitral annular area back to its normal state, but that the mitral annular excursion was not changed. This study shows that 3D echocardiography can be applied to animal studies to properly evaluate new techniques and devices more accurately than 2D echocardiography. In addition to the metrics they could only measure with 3D echocardiography, they were also able to visually evaluate changes in the 3D mitral annular geometry[12].

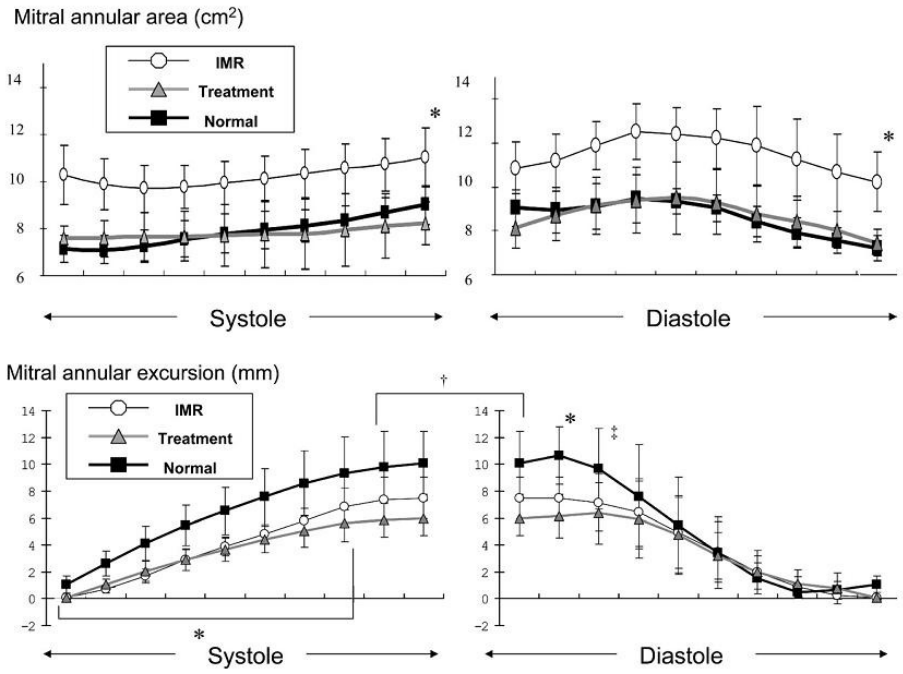


Figure 2.15 Impact of percutaneous mitral annuloplasty on dynamic mitral annular area and mitral annular excursion[12].

3D echocardiography has also been used to gain a better understanding of how the annular dynamics change under different disease conditions. Little et al investigated the dynamic annular geometry and function in patients with functional (F-MR) and prolapse mitral regurgitation (P-MR). With 3D echocardiography and certain software they were able to dynamically track the 3D shape, displacement, and area changes in the mitral annulus of various patients (Figure 2.16). Their results showed that if the dynamic area change and displacement of patients with functional and prolapsed mitral regurgitation differed from normal patients in different ways (Figures 2.17). The annular area in patients with P-MR was further from normal than those with F-MR. However, the percentage area change over the cardiac cycle and the maximum annular displacement in patients with F-MR was further from normal than those with P-MR. This indicates that while both pathologies cause mitral regurgitation, their mechanisms differ and therefore so should their treatments.

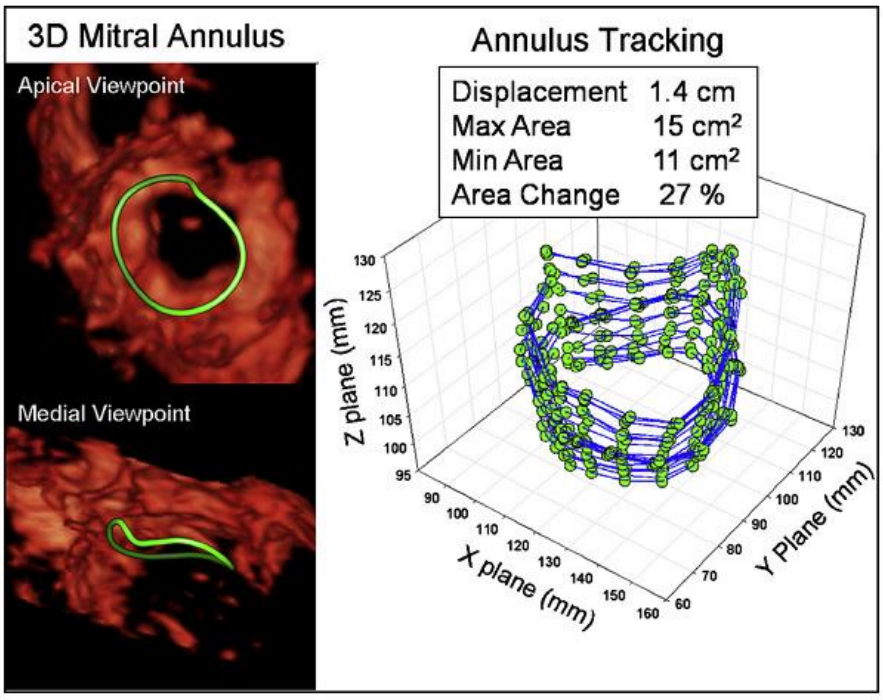


Figure 2.16 Tracking of 3D annular shape throughout the cardiac cycle[13].

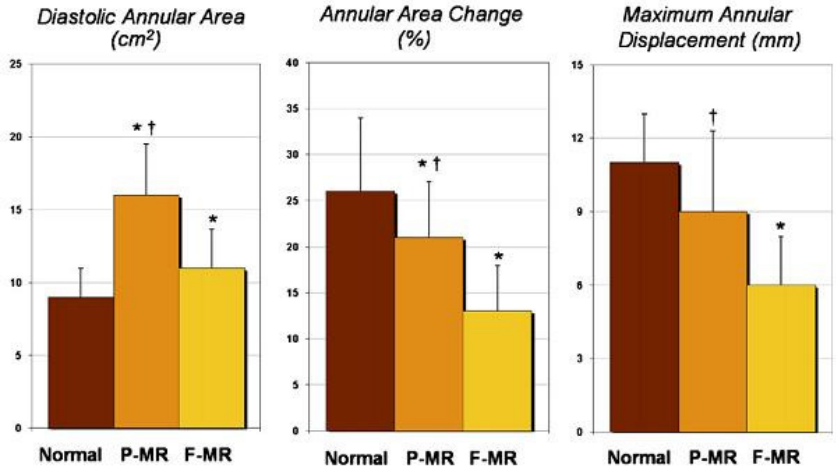


Figure 2.17 Results from Little et al showing the changes in distolic annular area (left), percentage area change (middle), and maximum annular displacment (right) for normal patients and those with functional mitral regurgitation (F-MR) and prolapse mitral regurgitation (P-MR).

Veronesi et al quantified the mitral apparatus dynamics in normal and diseased cases using real-time 3D echocardiography in 2008. They examined multiple parameters in normal, ischemic mitral regurgitation, and dilated cardiomyopathy cases. Two

dynamic measurements were of the mitral annulus longitudinal motion and the mitral annulus surface area (Figure 2.18). In addition, the mitral annulus velocity was quantified for normal, ischemic, and dilated cardiomyopathy (Figure 2.19). Their research showed that there was a significant difference between the maximum annulus displacement, peak systolic velocity, peak early diastolic velocity, and peak late diastolic velocity for ischemic and dilated cardiomyopathy patients compared to normal (Figure 2.20)[14].

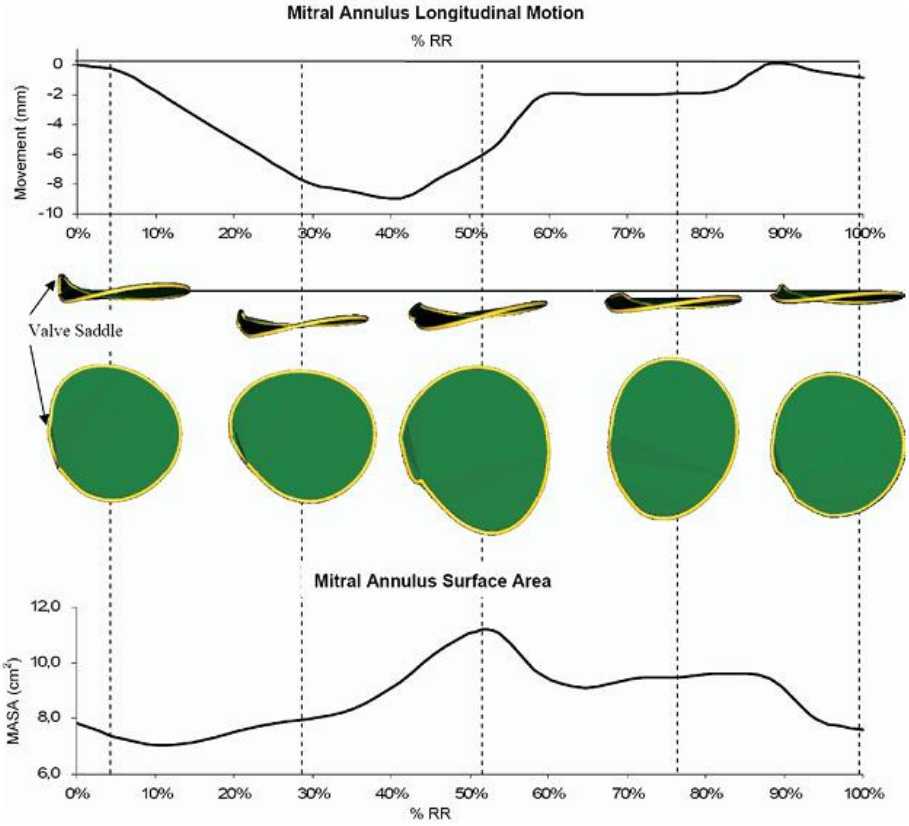


Figure 2.18 Dynamic mitral annulus longitudinal motion and 3D surface area over the cardiac cycle for a normal patient[14].

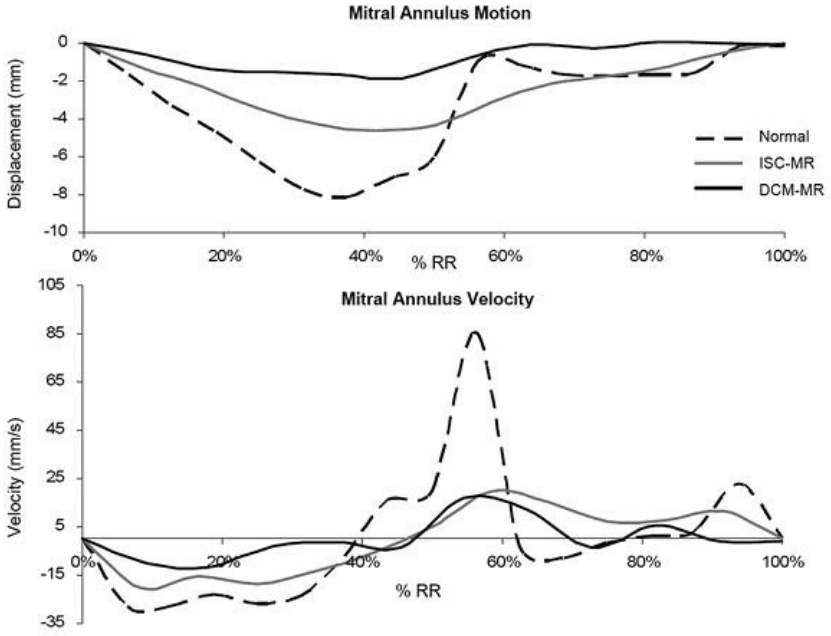


Figure 2.19 Mitral annulus motion and velocity for normal, ischemic, and dilated cardiomyopathy cases[14].

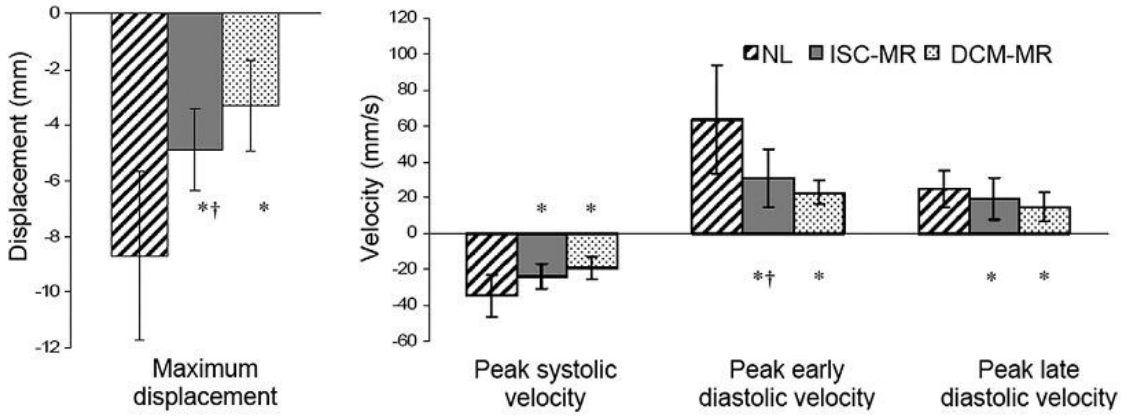


Figure 2.20 Maximum displacement, peak systolic velocity, peak early diastolic velocity, and peak late diastolic velocity for normal, ischemic, and dilated cardiomyopathy cases[14].

In 2008, Tsukiji et al demonstrated that 3D echocardiography could be used to quantify mitral valve coaptation in three dimensions, using 3D surface area instead of the normal one dimensional coaptation length used in 2D echocardiography (Figure 2.21). Leaflet tenting volume was also measured, which cannot be done with 2D echocardiography. The authors showed that there was a significant difference in multiple

parameters between normal and dilated cardiomyopathy patients. Specifically, the mitral annular area, leaflet tenting volume, and the leaflet coaptation surface area. It was noted that real-time 3D echocardiography allows for the measurement of new parameters that could better characterize the underlying causes of mitral regurgitation. It also allows for a better assessment of repairs to be made compared to 2D echocardiography[15].

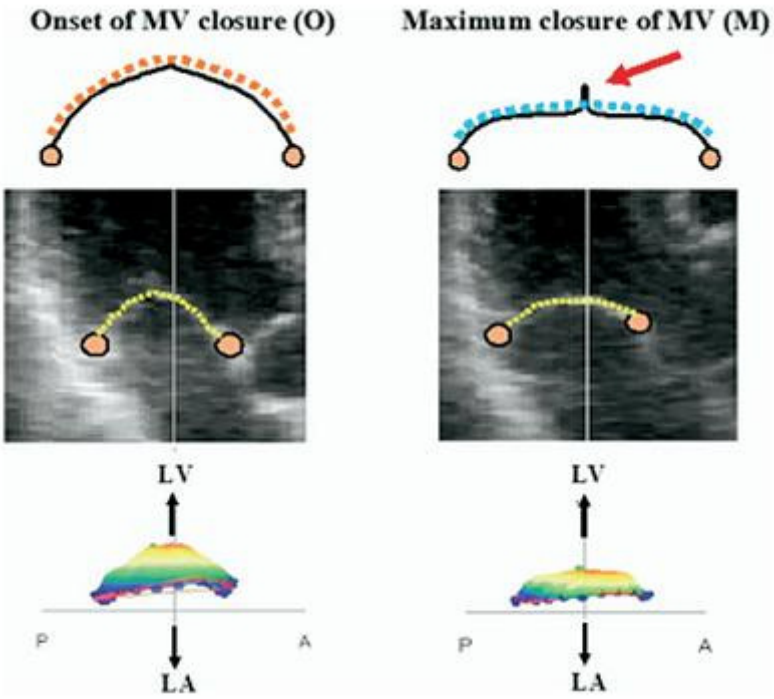


Figure 2.21 3D coaptation surface area measurements using real-time 3D echocardiography[15].

In the 2009 Echocardiography-Guided Interventions Guidelines and Standards from the Journal of the American Society of Echocardiography, it was noted that 3D imaging of the mitral valve using echocardiography can play a significant role in the planning of surgical or percutaneous approached to mitral valve repair. 3D echocardiography was also found to accurately visualize post-procedure commissural splitting and leaflet tears in patients undergoing percutaneous balloon mital

valvuloplasty. In addition, the role of 3D echocardiography to aid percunteous valve repairs was demonstrated (Figure 2.22)[16].

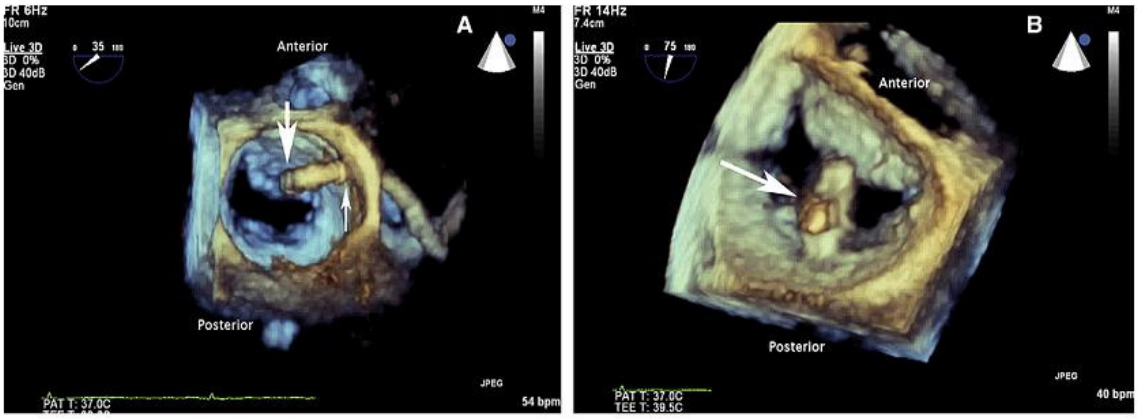


Figure 2.22 3D transesophageal images of a percuaneous edge-to-edge repair for mitral regurgitation using a clip[16].

In 2010, Biaggi et al, examined the use of 3D echocardiography in understanding the anatomy, mechanism, and severity of regurgitation in a patient with a cleft posterior mitral valve. This pathology is rare and complex. Given this, it could be hard to properly diagnosis this pathology using conventional 2D echocardgraphy. The authors showed that 3D echocardiography is capable of capturing complex mitral valve anatomy and aid in the proper diagnosis. The surgical plan was made with full knowledge of the mitral regurgitation mechanism. This resulted in a successful mitral valve repair where the prolapsing segments were detached from the annulus, the calcified annular area was resected en bloc, the P2 and P3 segments were reinserted into the myocardium, the cleft was closed by interrupted sutures, the chords were replaced with polytretrafluoroethylene sutures, and a flexible posterior annuloplasty band was placed on the valve. Since the repair was complex, the details of the pathology aided the surgeons in planning a success a successful repair before the surgery instead of once they had surgically inspected the pathology.

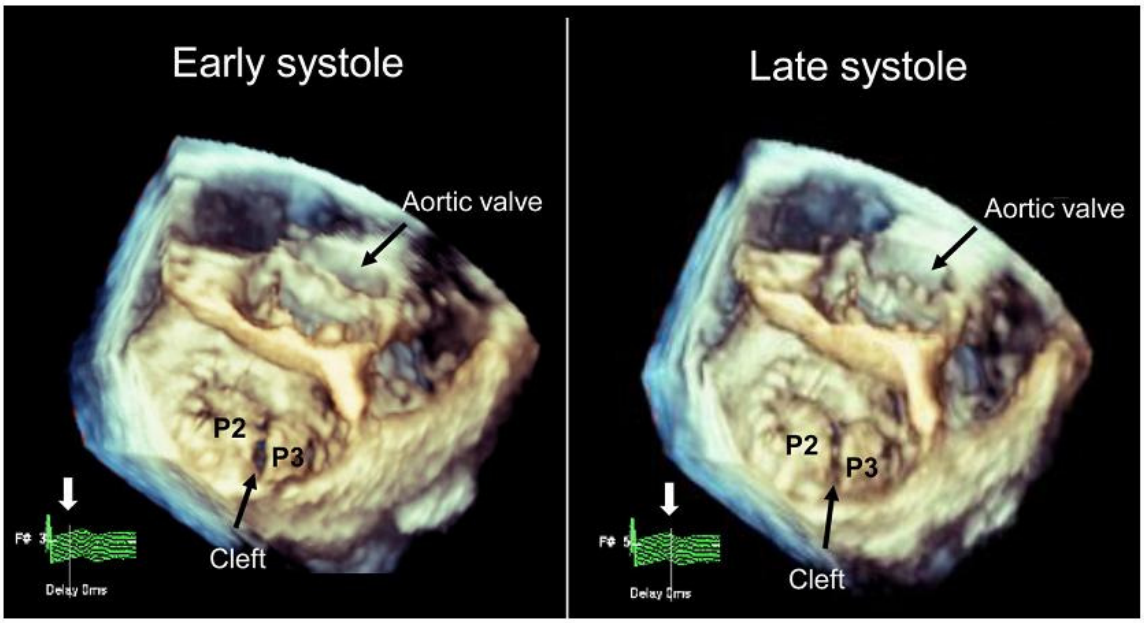


Figure 2.23 3D full volume acquisition of mitral and aortic valves in a patient with a cleft posterior mitral valve leaflet[17].

Recently, Masffessanti et al used 3D echocardiography intraoperatively to quantify the mitral apparatus in mitral prolapse cases before and after annuloplasty. The authors examined normal, fibroelastic deficiency (FED), and Barlow's disease (BD) cases. The area of each leaflet was measured before and immediately after the repair. In addition, the coaptation area was measured. It was demonstrated that the repairs significantly improved the planarity index, leaflet area, and the coaptation area. All measurements moved closer to the control measurements after repair (Figure 2.24)[18].

Variable	Controls	FED		BD	
		Before	After	Before	After
Planarity index (%)	12.7 ± 3.4	9.3 ± 3.6*	10.3 ± 5.3	10.5 ± 3.3*	11.8 ± 5.0
Leaflet measurements					
A3DE (cm2)	8.8 ± 2.4	15.0 ± 4.1*	6.0 ± 1.7*‡	19.4 ± 5.5*†	7.3 ± 1.9*†‡
A3DE anterior (cm^2)	4.8 ± 1.4	7.1 ± 1.8*	3.8 ± 1.2*‡	8.9 ± 3.2*†	4.5 ± 1.3‡
A3DE posterior (cm^2)	4.0 ± 1.1	7.7 ± 2.8*	2.2 ± 0.8*‡	10.4 ± 2.9*†	2.8 ± 1.0*‡
Coaptation					
Area (cm^2)	1.8 ± 0.8	–	1.9 ± 0.6	–	2.5 ± 0.8*†
Height (mm)	6.3 ± 1.7	–	7.7 ± 1.6*	–	9.0 ± 2.2*†
Length (mm)	30 ± 5	–	24 ± 6*	–	28 ± 6
Aortic-to-mitral plane angle (°)	136 ± 12	141 ± 12	134 ± 14	137 ± 11	135 ± 13

Figure 2.24 Measurements obtained from 3D quantification of MV controls and patients with FED and BD immediately before and after MV repair[18].

2.4 Mitral Valve Segmentation

Medical image segmentation is an important problem in the clinical realm today. It has enabled clinicians to extract quantitative and qualitative data from medical images to improve diagnosis and plan surgery. While there are many methods to segment of blood vessels and aneurysms, there has not been a rapid development of automated tools to segment dynamic mitral valve leaflets from 3D echocardiography images. This is most likely because of the complex structure of the leaflets compared to blood vessels and the lower quality and resolution of echo data compared to the MRI or CT scans normally used in vessel segmentation. Segmentation methods typically fall into three categories: manual, semi-automatic, and automatic. Manual methods rely entirely on human input to define the segmentation, semi-automatic methods require human input to guide the automatic portion of the segmentation, and automatic methods do not require human input to run.

In 1998, Mikic et al. presented a segmentation scheme for 2D echocardiography based upon active contours[19]. This algorithm, while automated, was only able to segment the anterior leaflet and sometimes required manual intervention (Figure 2.12). In addition, the mitral annulus was not tracked as part of their algorithm.

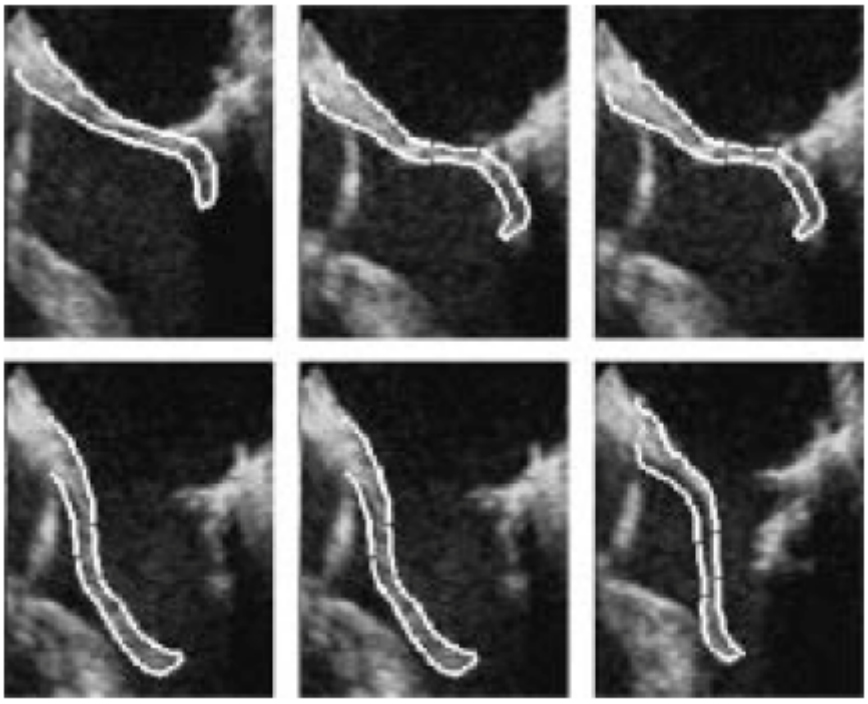


Figure 2.25 Tracking of the anterior mitral valve leaflet boundary. Manual correction was required for the upper right frame[19].

In 2004, Bashein et al. presented a study in which mitral leaflet surface models were generated with a manual segmentation method that used Bezier curves[20]. The objective of this study was to validate the reconstruction of the mitral leaflets using an *in vitro* porcine model. The surfaces generated from 5-degree and 10-degree rotational ultrasound scans were compared to laser scans of casts of the atrial side of the leaflets. Ten porcine mitral valves were evaluated with the ultrasound reconstructions and laser casts having a mean absolute deviation of 0.65 ± 0.12 mm for the 5-degree rotational scans and 0.64 ± 0.12 mm for the 10-degree rotational scans. Figure 2.13 depicts an example valve from this study.

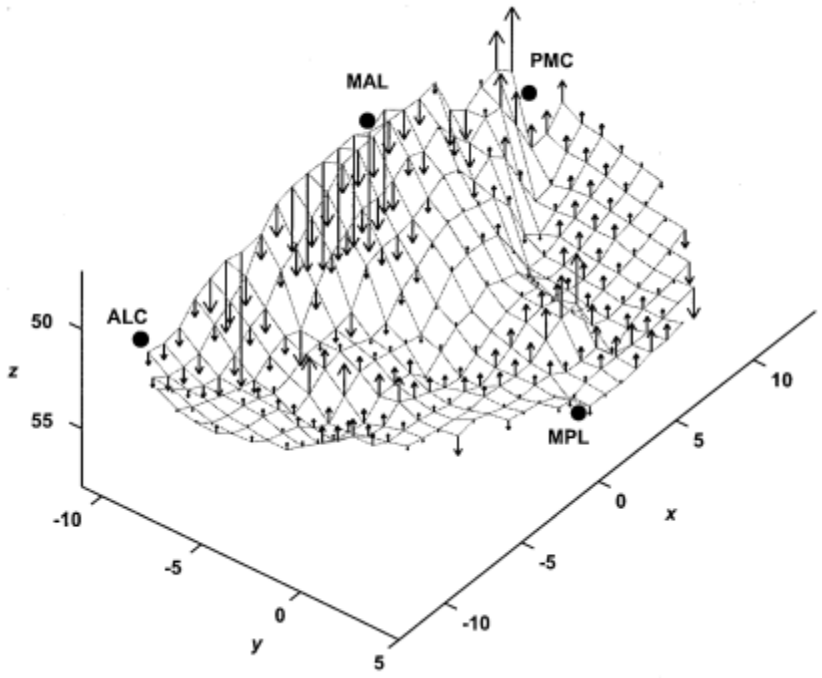


Figure 2.26 Example wire mesh generated using the method presented by Bashein et al. The arrows represent the deviation (distance) between the reconstruction and the laser scan[20].

The example reconstruction shown in Figure 2.14 resulted in a mean absolute deviation of 0.83 mm and the 95th percentile at 1.94 mm. While this study concluded that 5-degree rotational scanning enabled accurate modeling of mitral leaflet surfaces *in vitro*, there was no attempt to distinguish between the anterior leaflet, posterior leaflet, or the coaptation region. In addition, only normal valves under static conditions were evaluated. This was a limitation of the method because the ventricle was filled with a plastic material that required hardening time prior to imaging and casting. Once hardened the leaflets were in a fixed position. While the authors attempted to create a distorted geometry for comparison, there was too much regurgitation to allow the plastic to cure.

In 2006, Martin et al. presented a semi-automatic method for tracking the mitral valve leaflet in transesophageal echocardiography[21]. The main objective was to track the mitral leaflet after a manual initialization using two constrained active contours and

curve fitting. There were drawbacks to the method presented. Firstly, the tuning of parameters would require the user to adjust the parameters until a "good" segmentation of the leaflet was obtained. In addition, the segmentation method did not segment the entire leaflet throughout the cardiac cycle and only segmented the anterior leaflet (Figure 2.15).

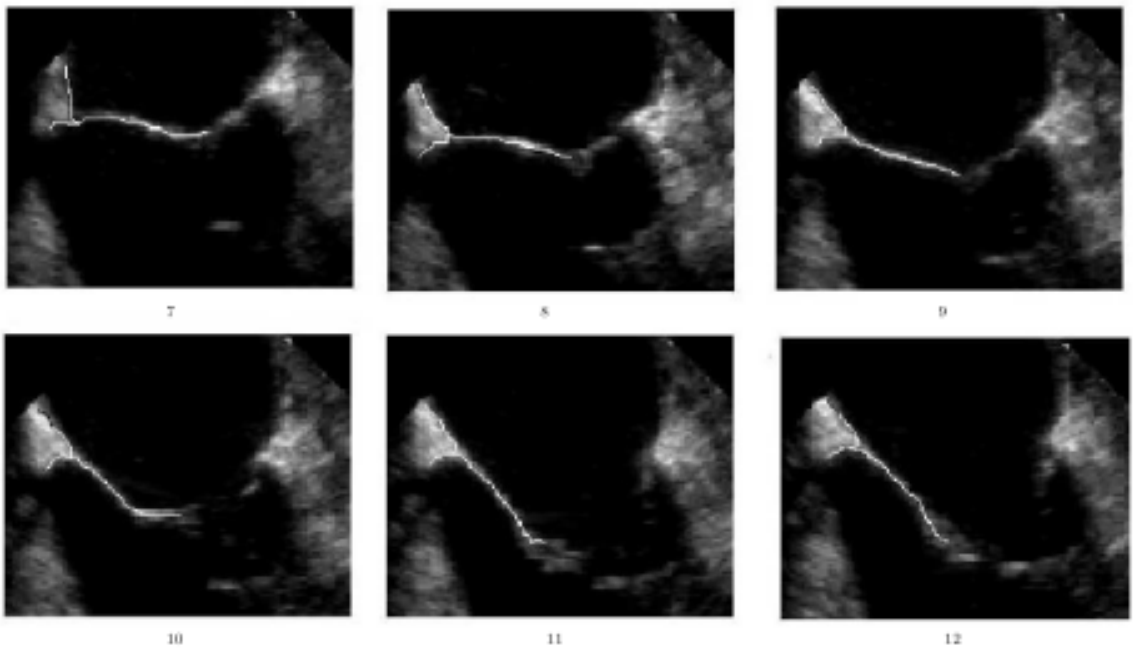


Figure 2.27 Sequence of segmentation images from the semi-automatic method presented by Martin et al. Note the incomplete segmentation of the anterior leaflet in the lower images[21].

In 2008, Shang et al. presented a new segmentation method called region competition based active contour[22]. This automated method was created for segmenting different types of medical images, including MRI, CT, and ultrasound. The developed method was applied to a 3D echocardiography image of a mitral valve (Figure 2.16). While this did result in a 3D reconstruction of the echocardiography image, it did not delineate the annulus from the leaflets or the leaflets from each other when the valve was closed.

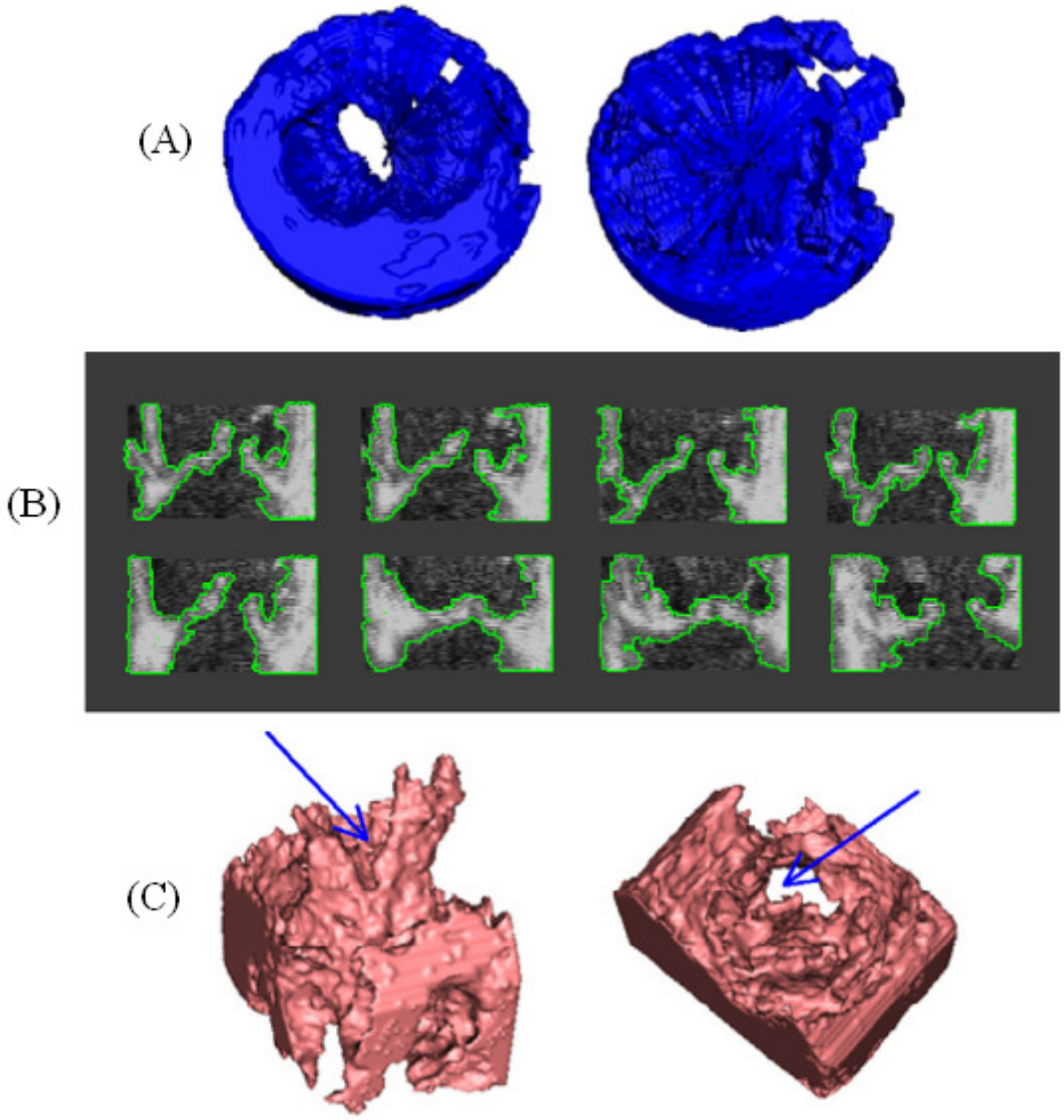


Figure 2.28 Segmentation of a 3D echocardiography data set of the mitral valve using the method developed by Shang et al. (A) Raw echocardiography data set. (B) Segmented 2D images of the mitral valve. (C) Model generated after segmentation[22].

In 2008, Ryan et al. presented a method for assessing human mitral leaflet curvature using RT3DE[23]. In this paper the annulus, anterior leaflet, and posterior leaflet were manually segmented from RT3DE images using free-hand curves. After segmentation the points generated by the segmentation were meshed using a MATLAB script and then smoothed. The segmentation was completed on the atrial surface of the leaflets as shown in Figure 2.29.

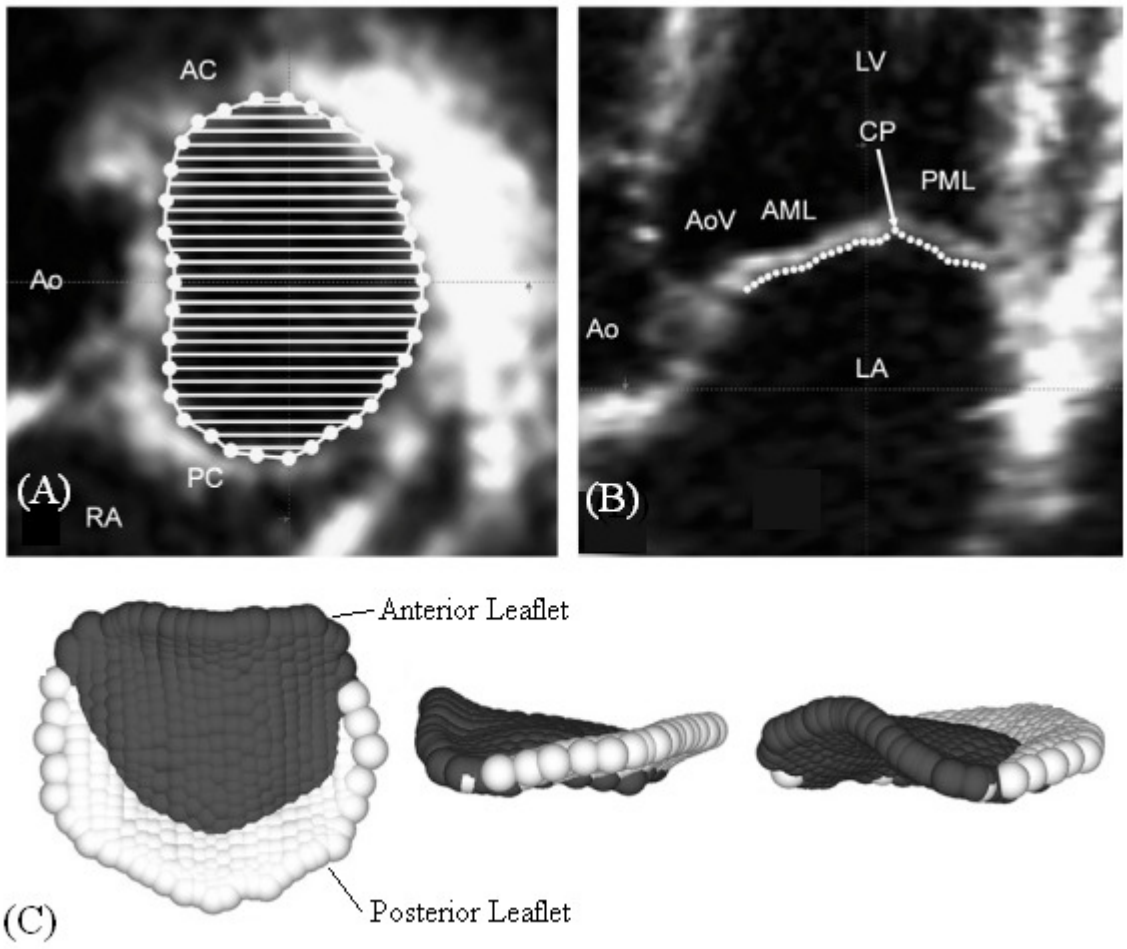


Figure 2.29 Segmentation of the mitral valve atrial surface. (A) Segmentation slices across the valve. (B) Segmented points across a single slice. (C) Segmented points for the anterior (dark grey) and posterior (white) mitral leaflets[23].

In 2009, Sneider et al. presented a method for the semi-automatic segmentation of the mitral annulus from 3D ultrasound[24]. Their method used a single user-specific point and a combination of graph-cut and active contour techniques to segment the 3D shape of the mitral annulus. While this method was able to segment the mitral annulus in a semi-automated fashion, it did not provide a method to capture the leaflet geometry. Also, the algorithm was not shown to be effective at tracking the dynamic annulus, only at a single time point.

Although methods exist to segment parts of the mitral valve, these focus on the valve during peak systole. Currently, there is no complete solution to segment the annulus and both leaflets throughout the cardiac cycle. This tool would be able to give the surgeon additional information about the dynamic structure of the valve prior to and after surgery. This tool would also be able to provide leaflet area and valve curvature measures throughout the cardiac cycle. This could aid in surgical planning and assist in determining the effectiveness and durability of a repair.

2.5 Clinical Significance

The advantage of using 3D echocardiography to aid in diagnosis and surgical planning has been recognized for over 20 years. In Levine et al[4], a spark gap assembly was used to take multiple 2D echocardiography images of the mitral valve and reconstruct them into a 3D volume. The objective was to characterize the 3D structure of the mitral valve as it relates to mitral valve prolapse. This was because the mitral valve has a 3D shaped annulus and the diagnosis of prolapse, which is leaflet displacement above the annulus, was dependent on analyzing the entire 3D structure of the valve. Specifically, a single 2D plane would likely not provide the proper diagnosis of mitral valve prolapse. One conclusion from this study was images acquired from 3D echocardiography would improve the diagnosis of mitral valve prolapse. This was because in the clinical setting the frequency of finding prolapse in the four-chamber view was related to the diligence with which it was searched for by the echocardiographer.[4] With the advent of real-time 3D echocardiography in recent years, there have been additional studies on the impact of 3D echocardiography to assess mitral valve morphology.

In 2003, Macnab et al[25] compared 2D multi-plane and 3D echocardiography to assess regurgitant mitral valve morphology. Using surgical findings were used as the gold standard, the study found that 3D echocardiography was significantly better than 2D

transesophageal echocardiography at delineating mitral valve segments and determining regurgitant valve morphology. They noted that 3D reconstruction is particularly useful in complex valves with large amounts of flail or redundant leaflet tissue. In these cases, the precise orientation of the imaging plane to the valve closure line was found to be invaluable in indentifying the underlying valvular abnormality.[25]

Valocik et al[26] examined the use of 3D echocardiography in mitral valve disease in 2005. Valocik and his collaborators found that 3D echocardiography was able to evaluate many mitral valve parameters that 2D echocardiography could not, including mitral valve area, 3D annular shape, and 3D regurgitant jet volume quantification. They noted that 3D echocardiography could capture the non-planarity of the mitral annulus and that this has become the gold standard for evaluating mitral structure. It has even introduced new criteria that have improved the diagnosis of mitral stenosis. Valocik et al also discussed how leaflet and papillary muscle geometry could be evaluated using 3D echocardiography and how this could be used to aid mitral valve repair surgical procedures.[26]

In 2008, Adams et al examined using 3D echocardiography to aid in surgical planning for degenerative mitral valve cases. It was concluded that systematic echocardiographic assessment and documentation is essential for guiding decisions on mitral valve surgery. In addition, a detailed assessment could maximize mitral valve repair rates because the valve morphology can be more easily determined with 3D echocardiography than 2D echocardiography. Specifically in determining whether a patient has a ruptured chordae tendineae with prolapse or excess leaflet tissue due to Barlow's disease. Adams also discussed how 3D echocardiography should be used to evaluate the valve before and after surgical repair to evaluate repair performance.[27]

Salcedo et al demonstrated how transesophageal 3D echocardiography (3DTEE) could be used with a framework to systematically characterize the mitral valve. They reviewed multiple papers in which 3DTEE was compared to 2D echocardiography and in

all studies 3D was found to be superior in characterizing geometry of the mitral valve. Figure 2.18 shows the identification of the scallops of the mitral valve using a 3D echocardiography image which would not be possible using 2D echocardiography. They also demonstrated that software could be used to generate representations to help aid the surgeons in understanding the valve morphology before surgery (Figure 2.19).[28]

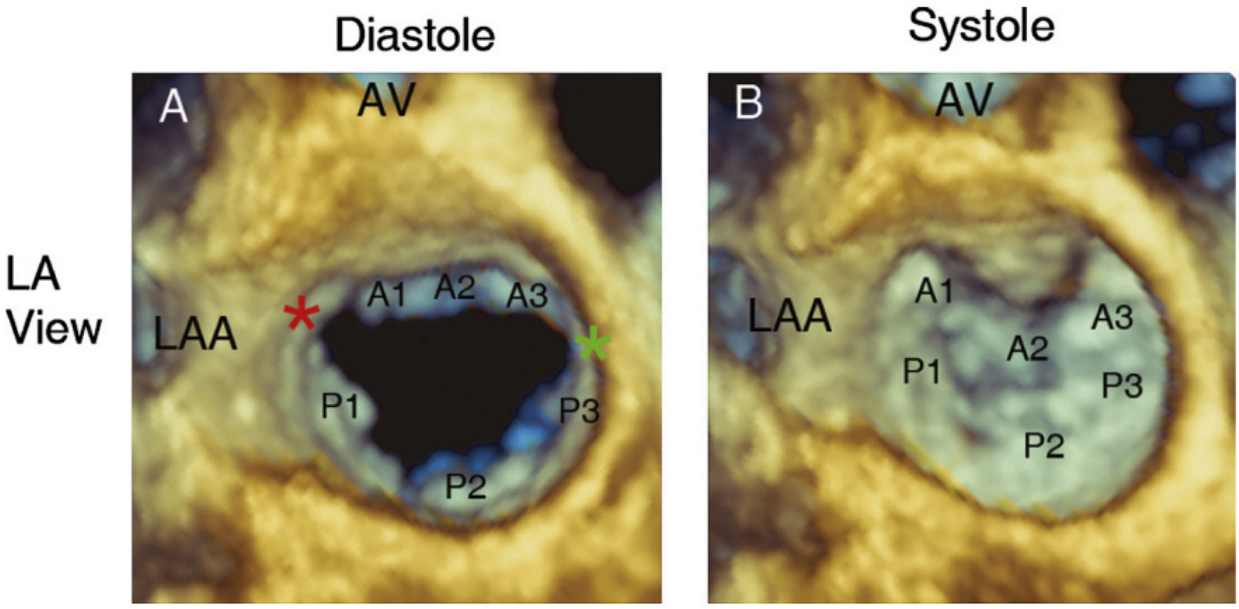


Figure 2.30 3DTEE image of the mitral valve[28].

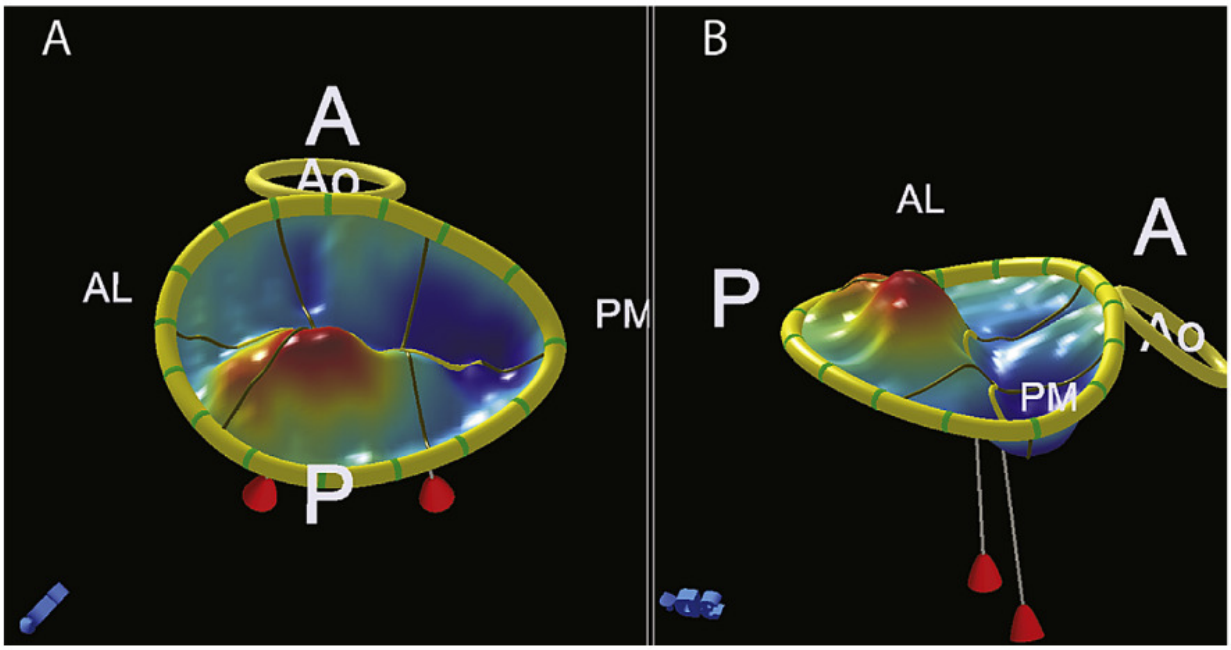


Figure 2.31 Representation of mitral valve annulus and leaflets generated from 3DTEE images[28].

While the clinical community agrees 3DE already improves surgical planning and determination of mitral valve morphology, there has not been a platform created for planning and simulation mitral valve surgery. There is also not a platform to quantify leaflet and annular motion throughout the cardiac cycle to better understand each valve. Such a tool would be useful in aiding clinicians in their decision making and take steps toward allowing surgical simulation prior to surgery. This could improve the diagnosis of valve morphology and improve surgical outcomes, which as a result, could improve patient outcomes.

CHAPTER 3

HYPOTHESIS AND SPECIFIC AIMS

3.1 Hypothesis

Image segmentation and interpolation techniques can be used to develop a tool to yield dynamic, valve specific virtual leaflet models of the mitral valve from 3D echocardiography images.

3.2 Specific Aims

The objective of this thesis is to use image segmentation techniques to develop and validate a code to generate virtual mitral valve models from 3D echocardiography data. To achieve this objective, the study is divided into three specific aims.

- In the first aim, a manual segmentation tool that generates virtual mitral valve leaflet models will be developed for use with 3D echocardiography.
- The second aim will be to validate the virtual mitral valve leaflet models by using dual camera stereo photogrammetry on a dynamic *in vitro* mitral valve model.
- The third aim will focus on investigating mesh refinement methods to decrease the time required to segment valve leaflet geometry and determine the method that best interpolates the mitral leaflets.

3.2.1 Specific Aim 1

Develop a robust tool to generate virtual leaflet models from real-time 3D echocardiography data. Current manual methods for generating mitral valve geometry from 3D echocardiography data create a single surface representing both leaflets of the mitral valve during peak systole. In order to obtain a more accurate representation of

mitral valve leaflet geometry, a manual segmentation tool will be developed to create dynamic virtual models of each leaflet throughout the cardiac cycle. Tools to scale, orient, and crop 3D echocardiography data sets will be developed to account for variability in mitral valve position and orientation in acquired data. The dynamic virtual model will be comprised of triangular surface meshes of the anterior and posterior leaflets.

3.2.2 Specific Aim 2

Validate the virtual mitral valve leaflet models with a dynamic in vitro mitral valve model. The virtual leaflet models will be validated using dual camera stereo photogrammetry as a gold standard. Dual camera stereo photogrammetry is an established imaging technique that can provide a 3D reconstruction of the mitral valve using two planar images. The transformation between the photogrammetry and echocardiography coordinate systems will be determined using a reference cube in the region of interest. A scheme to account for errors in the *in vitro* 3D ultrasound measurement will be developed. The expected deviation between the 3D valve reconstruction from photogrammetry and the virtual model surface will be determined and the actual deviation between the 3D reconstruction and the virtual model will be calculated. If the error is greater than expected, then the sources of error will be examined to determine the cause of the mismatch. The marker data will also be fitted to the virtual model using a best fit algorithm in order to compare the shapes of the 3D marker reconstruction and the virtual model.

3.2.3 Specific Aim 3

Investigate interpolating triangular mesh refinement methods to decrease segmentation time and determine the method that best interpolates the mitral leaflets using a dynamic in vitro model. A mesh refinement method begins with a polygon mesh

and then subdivides it by adding new vertices and faces. New vertices are based upon the coordinates of nearby original vertices and for interpolating refinement schemes the vertices of the original mesh remain stationary in the newly generated polygon mesh. To reduce the time required to complete the manual segmentation, interpolating triangular mesh refinement methods will be investigated. Mesh refinement stopping criteria will be determined and the ability of each method to interpolate the leaflet will be assessed using the 3D reconstructions from the dynamic *in vitro* model and the same deviation measurements used in specific aim 2.

CHAPTER 4

MATERIALS AND METHODS

4.1 Segmentation Method Development

Segmentation methods are generally divided into three regimes: manual, semi-automated, and automated. Manual methods require the user to define the entire segmentation. Semi-automated methods require an initial user input to guide the automated segmentation. Automated methods do not require user input to guide the segmentation, but may require parameter adjustments to automatically segment the area of interest.

Current efforts to segment mitral valve geometry from echocardiography images have included manual, semi-automated, and automated schemes. Philips® QLAB Mitral Valve Quantification Tool uses a manual approach, and produces a single mesh for both leaflets at peak systole. Semi-automated methods have focused on segmenting the anterior leaflet from 2D echocardiography images, and still require manual correction in some instances[19, 21]. An automated approach has been developed for medical images and applied to the 3D echocardiography of the mitral valve[22]. However, this method does not delineate the different features of the mitral valve.

To improve upon current methods, this study sought to develop a manual method capable of segmenting, and delineating the annulus and both leaflets during the entire cardiac cycle. Manual segmentation by an expert observer has been used as the standard for evaluating semi-automated and automated segmentation methods[19, 22]. In addition, a manual method may allow for the amount of post-segmentation mesh processing required

to be reduced. While there are advantages to a manual segmentation method, there are drawbacks such as user variability and increased segmentation time.

A manual segmentation tool with the ability to segment the mitral leaflets through time must be able to do the following:

1) Import 3D echocardiography data

2) Scale, rotate, and crop 3D echocardiography data to account for variability in valve position between data sets

3) Select or generate points along the leaflets and annulus based upon user-input

4) Create connectivity between the segmentation data to generate a surface representation of the leaflets

Once these requirements have been satisfied, the tool must be validated. In addition to validation, methods to minimize the drawback of high segmentation time should be investigated. Reducing segmentation time could be achieved through mesh refinement methods. Mesh refinement methods generate new vertices in a mesh based upon the original vertices. A flow chart of how the tool could work with the aforementioned requirements is depicted in Figure 4.1.

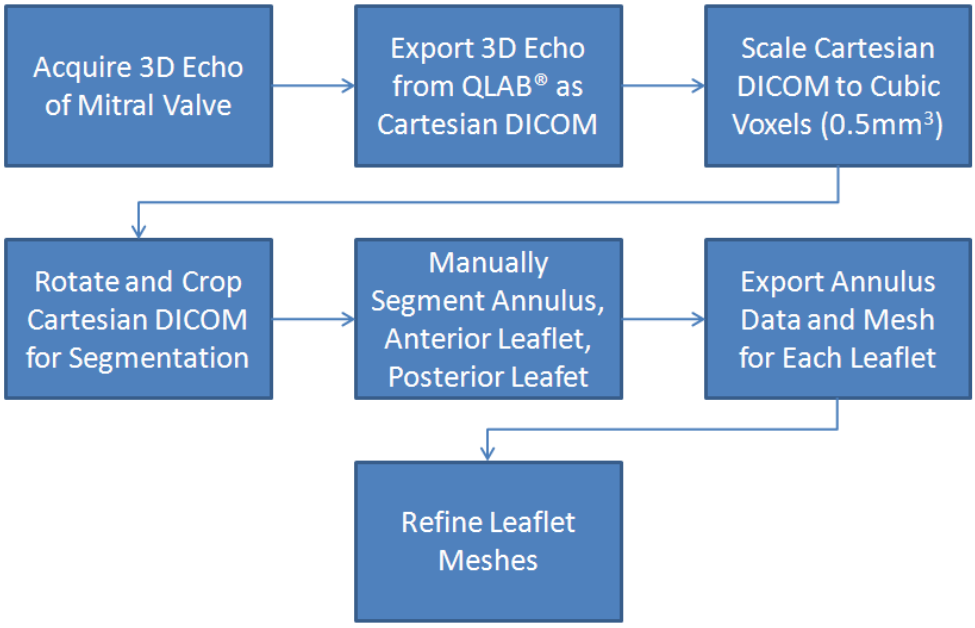


Figure 4.1 Segmentation tool flow chart.

To import the 3D echocardiography data into the segmentation tool, Philips Medical Systems (Andover, MA) QLAB software was used to convert a 3D echocardiography image with spherical coordinates to a Cartesian DICOM. The Cartesian DICOM provided an image stack that was imported into MATLAB (Natick, MA). To account for variability in valve position, methods for scaling, rotating, and cropping the Cartesian DICOM were developed in MATLAB. The third requirement of segmenting the valve based upon user input was addressed by investigating multiple input methods in MATLAB. These methods included using points, lines, and curves to select the leaflets. To satisfy the requirement of creating a mesh from the segmentation data, methods to triangulate the segmentation data were investigated and implemented in MATLAB.

Once all the methods to satisfy the tool requirements were developed, a method to validate the virtual models generated by the tool was developed. After validation,

multiple methods of mesh refinement (Loop, Butterfly, Interslice) were investigated to reduce segmentation time. These methods were compared to determine which best interpolated the leaflet geometry. The final segmentation tool satisfied all of the requirements and was able to generate virtual leaflet models for the entire cardiac cycle.

4.1.1 Cartesian DICOM Conversion

3D echocardiography data is formatted four-dimensionally in spherical spatial coordinates and time. QLAB (Philips Medical Systems) software was used to convert the native 3D echocardiography data into a four-dimensional data set in Cartesian spatial coordinates and time. After processing, the resulting Cartesian DICOM could be imported into MATLAB for subsequent user defined operations.

4.1.2 Segmentation Methods

Point, line, and curve based manual segmentation methods were investigated. For each of these methods, the same segmentation process was used. First, a method to import 2D images into the tool was developed. After this the 2D segmentation slices were displayed and segmented by the user. Then the user would move through the image stack 3 slices (1.5mm) at a time to display and segment each 2D image across the valve as illustrated in Figure 4.2.

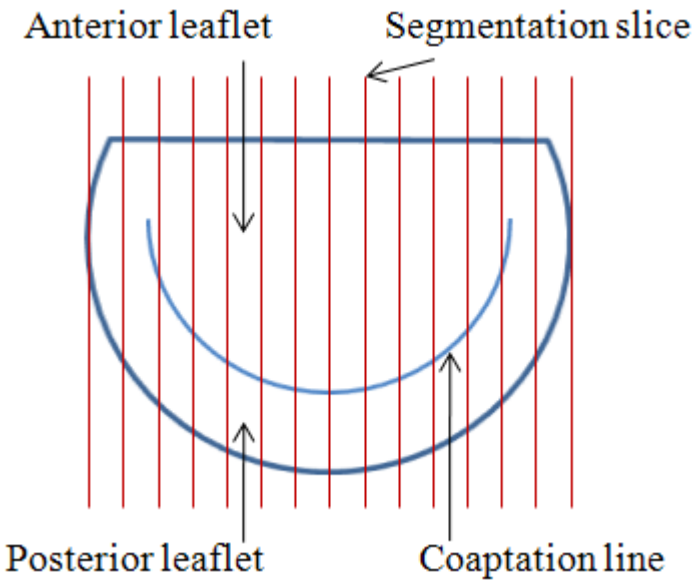


Figure 4.2 Diagram of segmented images (slices) across the mitral valve.

Once the segmentation was complete, the segmented points were triangulated and a mesh for each leaflet was generated and exported.

4.1.3 Point Segmentation

The first method of manual segmentation investigated was the selection of unconnected points on the mitral annulus, anterior leaflet, and posterior leaflet. After the Cartesian DICOM was exported from QLAB, a MATLAB script generated a set of 2D anterior-posterior mitral valve images from the Cartesian DICOM in the commissure-commissure direction for a single time point (Figure 4.2). The developed tool allowed the user to move through the image stack and segment each 2D image by selecting points. An example of the graphical user interface (GUI) for this method is pictured in Figure 4.3.

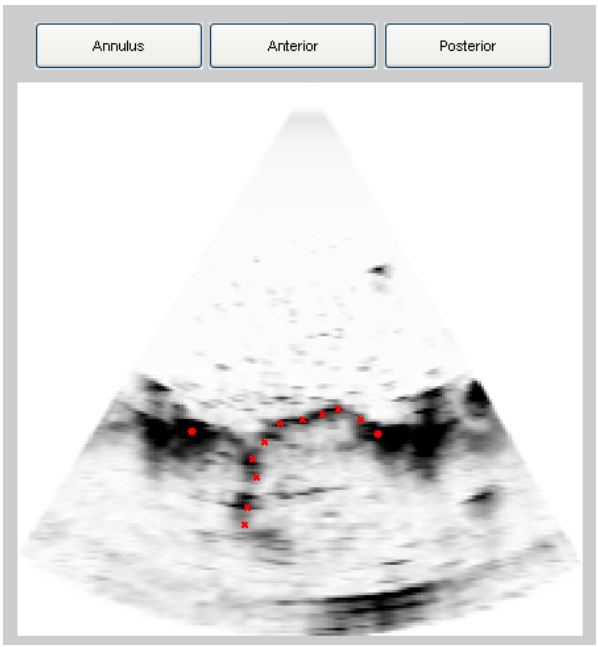


Figure 4.3 Graphical user interface (GUI) for point segmentation method. The annulus is represented by the red dots and the points chosen for the anterior leaflet are represented by red Xs.

The tool first required the user to select two annulus points. Then any number of points could be selected for the anterior and posterior leaflets. The image coordinates of every selected point were stored. The following is an example of the MATLAB script that was used to store and plot the user-selected points for the anterior leaflet:

```
i=0; % initialize index variable i (number of points selected)
button=1; % define button as pressed to enter while loop
while button==1
    [x,y,button]=ginput(1); % x,y and button number from button press
    x=round(x); % round for actual pixel value
    y=round(y); % round for actual pixel value
    if button==1
        i=i+1; % increase the number of selected points by 1
        meshData(s).anterior(i,:)=([s y x]); % store selected point
        px=meshData(s).anterior(:,3); % define x values to be plotted
        py=meshData(s).anterior(:,2); % define y values to be plotted
        plot(px,py,'rx','LineWidth',2); % plot red Xs for selected pts
    end
end
```

where:

i: number of points selected

button: button number last pressed

x: x-coordinate of the most recently selected point

y: y-coordinate of the most recently selected point

s: slice number of the image being segmented

meshData.anterior: storage variable for the selected points in a particular image

px: x-coordinates of all selected points

py: y-coordinates of all selected points

This script was initiated by the user pressing a button to segment the anterior leaflet to allow point selection. Each time a new point was selected, the *x* and *y*-coordinates were stored in *meshData.anterior*. Then, all selected points were plotted on the image. When all the points desired were selected, the user had to click either the middle or right mouse button. This changed the *button* variable to a value other than 1, and caused the segmentation *while* loop to end. The annulus points were selected using a similar MATLAB script that only allowed the selection of two points.

Once segmentation of each 2D image was complete, the 3D coordinates of each selected point were exported to text files for each valvular component (annulus, anterior leaflet, posterior leaflet). To create a mesh of each leaflet, the connectivity between the segmented points needed to be defined. The first method explored was the triangulate

function in the software Tecplot 360 (Bellevue, WA). The segmented points of each leaflet were imported into Tecplot 360 and triangulated. This method created a surface mesh of the leaflets that wrapped around on itself (Figure 4.4). It was determined that this happened because the triangulate method in Tecplot 360 ignores the z-coordinate of the data being triangulated. The triangulation could have also been affected by variability in the number of segmented points in each slice. Due to the poor triangulation generated, this method was not pursued further.

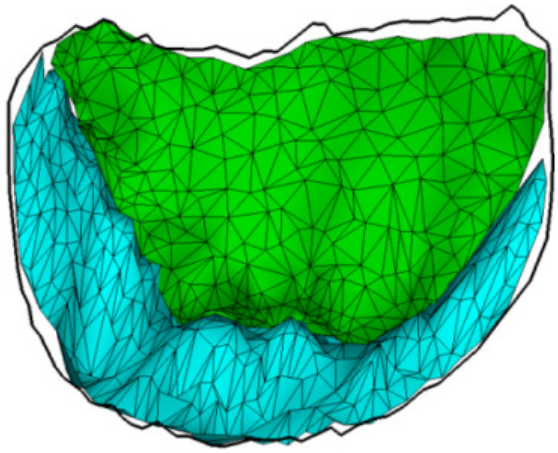


Figure 4.4 Tecplot 360 triangulation of the point segmentation data (annulus=black, anterior leaflet=green, posterior leaflet=blue).

Two areas of potential improvement to the point segmentation method were identified. The first was to help the user determine how many points were required to represent the leaflet geometry. The second was to improve the triangulation of the segmented points. To show the user how straight line connectivity of the selected points would represent the leaflet, a line based segmentation was adopted. To improve the triangulation, a method was developed to explicitly define the connectivity between neighboring segmented images.

4.1.4 Line Segmentation Method

The use of lines instead of unconnected points was investigated for the manual segmentation technique to eliminate some of the drawbacks from the previous method. The main improvement was that the user could select a series of ordered and connected points from the annulus to the free edge of each leaflet. This provided a qualitative metric of how well the selected points captured the leaflet geometry. The number of points the user could select in each image was not constrained, so each image could contain a different number of segmentated points. The following MATLAB code demonstrates how this was implemented for segmenting the anterior leaflet:

```
i=0; % initialize index variable i (number of points selected)
button=1; % define button as pressed to enter while loop
while button==1
    [x,y,button]=ginput(1); % x,y and button number from button press
    x=round(x); % round for actual pixel value
    y=round(y); % round for actual pixel value
    if button==1
        i=i+1; % increase the number of selected points by 1
        meshData(s).anterior(i,:)=([s y x]); % store selected point
        px=meshData(s).anterior(:,3); % define x values to be plotted
        py=meshData(s).anterior(:,2); % define y values to be plotted
        plot(px,py,'y','LineWidth',2)   % plot yellow line
        plot(px,py,'rx','LineWidth',2); % plot red Xs of selected pts
    end
end
```

where:

i: number of points selected

button: button number last pressed

x: x-coordinate of the most recently selected point

y: y-coordinate of the most recently selected point

s: slice number of the image being segmented

meshData.anterior: storage variable for the selected points in a particular image

px: x-coordinates of all selected points

py: y-coordinates of all selected points

Another improvement implemented was to import the entire Cartesian DICOM into the segmentation tool. This eliminated the step of exporting each time point to a stack of images and enabled the creation of a quad view GUI with the ability to display multiple planes from the same data set (Figure 4.5).

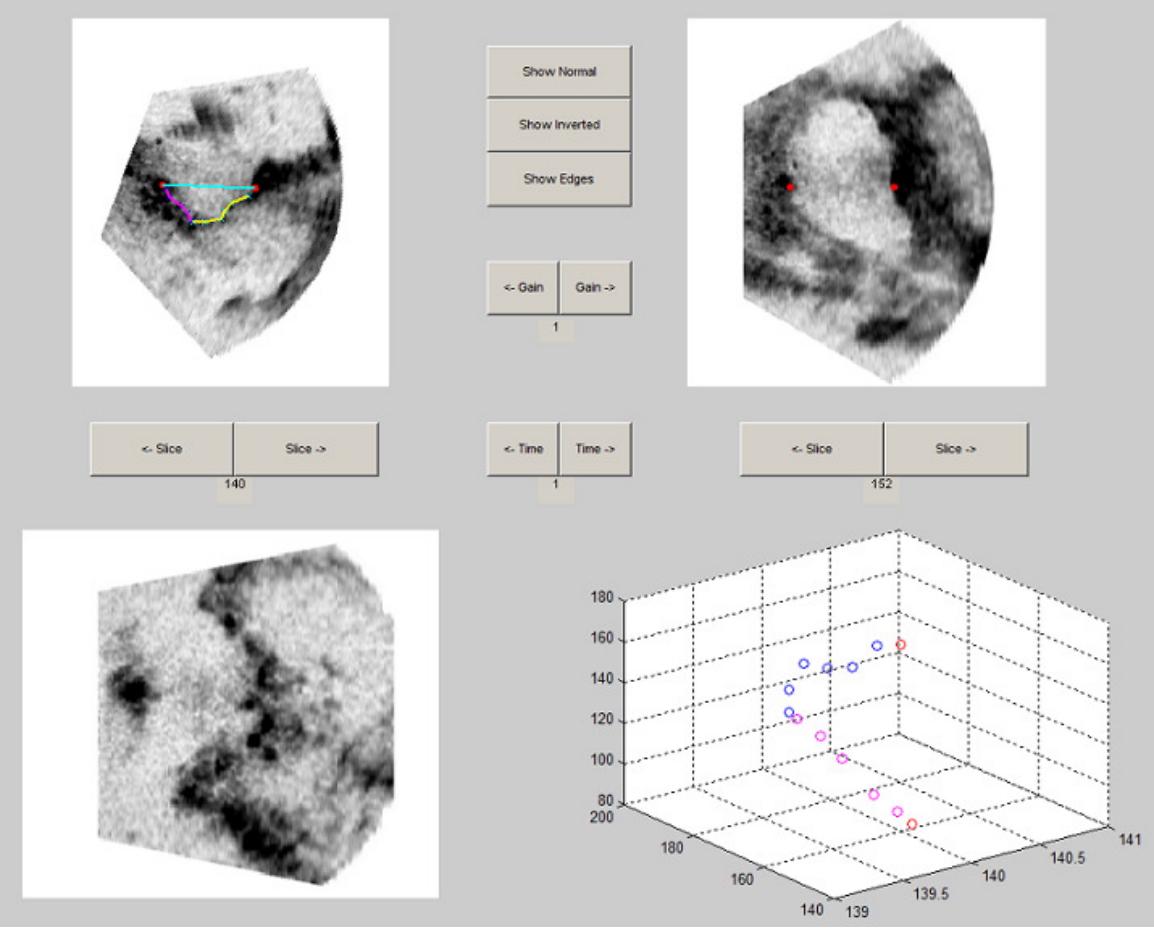


Figure 4.5 Quad view GUI from line segmentation method. Top-left: Anterior-posterior view of leaflets, top-right: plane of annulus, bottom-left: Commissure-commissure view of leaflets, bottom-right: 3D plot of segmented points.

The quad view GUI enabled the *xy*, *xz*, and *yz*-planes of the Cartesian DICOM to be displayed simultaneously. This provided the user with additional views of the data including the annulus plane, a commissure-commissure slice, and a 3D plot of the segmented data points. This also enabled the segmented annulus to be plotted on the annulus plane, which assisted the user segmenting the annulus.

In conjunction with other improvements, a new triangulation method was developed. The developed method only created connectivity between neighboring

images. This allowed the triangulation to be explicitly defined and avoided the computation cost and possible incorrect triangulations from using an algorithm for triangulation. Another advantage of explicitly defining the triangulations was that it allowed the triangles to be consistently oriented in the clockwise direction. For a given data set with slices 50, 53, 56, 59, and 62 segmented, the triangulation is first defined between 50 and 53. After this, each slice was incremented by the slice spacing (in this case three) and then the triangulation between those two slices was generated. In this example the triangulation would be generated in the following order: 50-53, 53-56, 56-59, and finally 59-62. Since the number of point between neighboring data sets could vary, the triangulation needed to be defined for three separate conditions:

1) Neighboring segmented data sets contain the same number of points

2) The first set of segmentation data contains fewer points than the second

3) The first set of segmentation data contains more points than the second

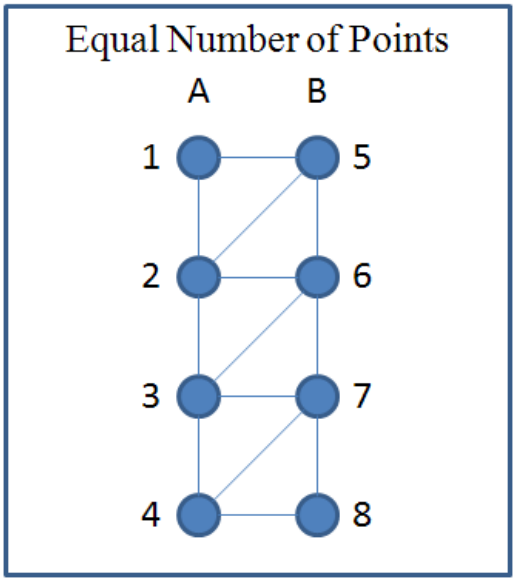


Figure 4.6 Example triangulation between two equal length data sets.

where:

A: indices of the points in the first segmentation data set (1,2,3,4)

B: indices of the points in the second segmentation data set (5,6,7,8)

The first condition is illustrated in Figure 4.6. Both the first (*A*) and second (*B*) sets of segmented points from neighboring images contain four (*n*) ordered points. For this case, the number of triangles generated will be 2*(*n*-1), so this example will generate 6 triangles. For the first pair of triangles between points 1, 2, 5, and 6, the triangulation generated will be [2 1 5] and [2 5 6]. For a generic pair of data set, pairs the triangles are defined as [*A*(*index+1*) *A*(*index*) *B*(*index*)] and [*A*(*index+1*) *B*(*index*) *B*(*index+1*)]. Here, *A*(*index*) and *B*(*index*) correspond to the point in each set of ordered points and *index* is defined from 1 to *n*-1.

The following MATLAB code demonstrates how the connectivity was formed for the equal length case:

```
A=[1 2 3 4];
B=[5 6 7 8];
n=length(A);
for index=1:1:n-1
    connectivity(2*index-1,:)=[A(index+1) A(index) B(index)];
    connectivity(2*index,:)=[A(index+1) B(index) B(index+1)];
end
```

where:

A: indices of the points in the first segmentation data set

B: indices of the points in the second segmentation data set

n: number of points in each data set

connectivity: indices of the resulting clockwise-oriented triangles between data set A and B

Figure 4.7 demonstrates an example triangulation for conditions 2 and 3.

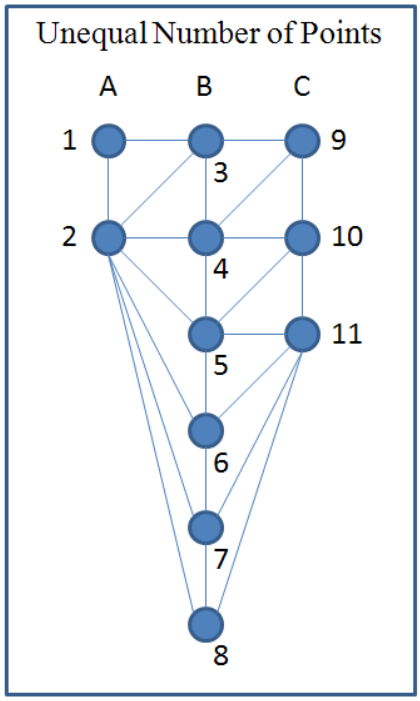


Figure 4.7 Example triangulation between unequal length data sets. The triangulation between A and B represents condition 2 and the triangulation between B and C represents condition 3.

where:

A: indices of the points in the first segmentation data set (1,2)

B: indices of the points in the second segmentation data set (3,4,5,6,7,8)

C: indices of the points in the third segmentation data set (9,10,11)

For both of these cases, the triangulation is the same as the equal length triangulation up to the length of the shorter data set. Each point past that length is connected back to the final point in the shorter data set. For the case between *A* and *B*, the clockwise triangulation is defined by the following MATLAB code where the length of the first set is smaller than the second (condition 2):

```
A=[1 2];
B=[3 4 5 6 7 8];
nA=length(A);
nB=length(B);
nDiff=nB-nA;
triCount=1;
for index=1:1:nB-1
    if index<nB-nDiff-1
        connectivity(triCount,:) = [A(index+1) A(index) B(index)];
        connectivity(triCount+1,:) = [A(index+1) B(index) B(index+1)];
        triCount=triCount+2;
    else
        connectivity(triCount,:) = [A(nA) B(index) B(index+1)];
        triCount=triCount+1;
    end
end
```

where:

A: indices of the points in the first segmentation data set (1,2)

B: indices of the points in the second segmentation data set (3,4,5,6,7,8)

nA: number of points in data set A

nB: number of points in data set B

triCount: index of the number of triangles currently defined in the for loop

connectivity: indices of the resulting clockwise-oriented triangles between data set A and B

The clockwise triangle connectivity between data sets *B* and *C* would be generated based upon the following MATLAB code, where the first set is longer than the second set (condition 3):

```
B=[3 4 5 6 7 8];
C=[9 10 11];
nB=length(B);
nC=length(C);
nDiff=nC-nB;
triCount=1;
for index=1:1:nC-1
    if index<nC-nDiff-1
        connectivity(triCount,:) = [B(index+1) B(index) C(index)];
        connectivity(triCount+1,:) = [B(index+1) C(index) C(index+1)];
        triCount=triCount+2;
    else
        connectivity(triCount,:) = [A(index+1) B(index) C(nC)];
        triCount=triCount+1;
    end
end
```

where:

B: indices of the points in the first segmentation data set (3,4,5,6,7,8)

C: indices of the points in the second segmentation data set (9,10,11)

nB: number of points in data set B

nC: number of points in data set C

triCount: index of the number of triangles currently defined in the for loop

connectivity: indices of the resulting clockwise-oriented triangles between data set A and B

The code for the condition 2 ($nA < nB$) differs from the condition 3 ($nB > nC$) by the *else* statement because the connectivity generated back to the last point in the shorter data set is different if the shorter data set is first or second.

By combining the methods for each condition, every set of data points was triangulated from one commissure to the other until a triangle mesh of the entire leaflet was created (Figure 4.8).

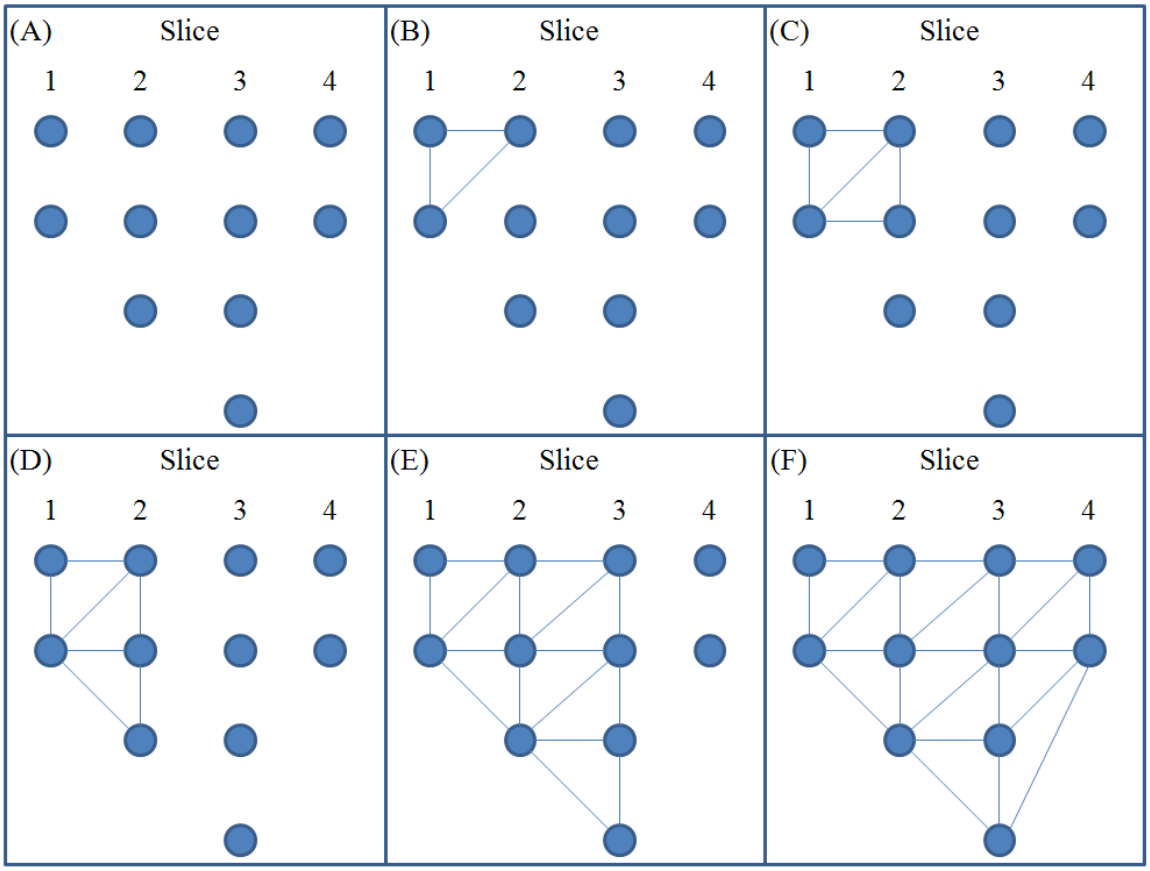


Figure 4.8 Triangulation steps across multiple segmentation data sets using the triangulation algorithm.

Once the triangulation for the entire segmentation was generated, a PLY file was written with vertex and connectivity information for use with Tecplot 360 and Geomagic

Studio 11 (Research Triangle Park, NC) software. PLY is a file format for the storage of graphical objects[29]. A typical PLY file has the following structure:

Header

Vertex List

Face List

(lists of other elements)

For the purpose of storing triangle meshes, only the header, vertex list, and face list were required. The following is an example of an ASCII description of a tetrahedron made up of triangles using the ASCII PLY format:

```
ply
format ascii 1.0           % ascii/binary, format version number
comment made by anonymous  % comments keyword specified, like all lines
element vertex 4           % 4 "vertex" elements in this file
property float32 x         % vertex contains float “x” coordinate
property float32 y         % vertex contains float “y” coordinate
property float32 z         % vertex contains float “z” coordinate
element face 4             % 4 "face" elements in this file
property list uint8 int32 vertex_index % "vertex_indices"=list of ints
end_header                 % delimits the end of the header
0 0 0                      % start of vertex list
1 0 0
0 1 0
0 0 1
3 0 1 2                    % start of face list
3 0 1 3
3 0 3 2
3 2 3 1
```

The element vertex row of the header defines the number of vertices in the vertex list. The element face row of the header defines the number of faces in the vertex list. Each row of the vertex list defines the *x*, *y*, and *z*-coordinate of that vertex. In each row of the face list, the first number defines the number of points in the face followed by the

vertex list index of each point in the face. The first point in the vertex list has an index of 0[29].

While the new triangulation method generated surface meshes for the mitral leaflets, poor triangulations occurred due to different numbers of points in neighboring slices (Figure 4.9). These poor triangulations can be characterized by triangles with acute angles (<5˚) and a lack of connectivity between the leaflets and annulus. The combination of these two problems can result in mesh artifacts and empty space between the leaflet and annulus (Figure 4.10).

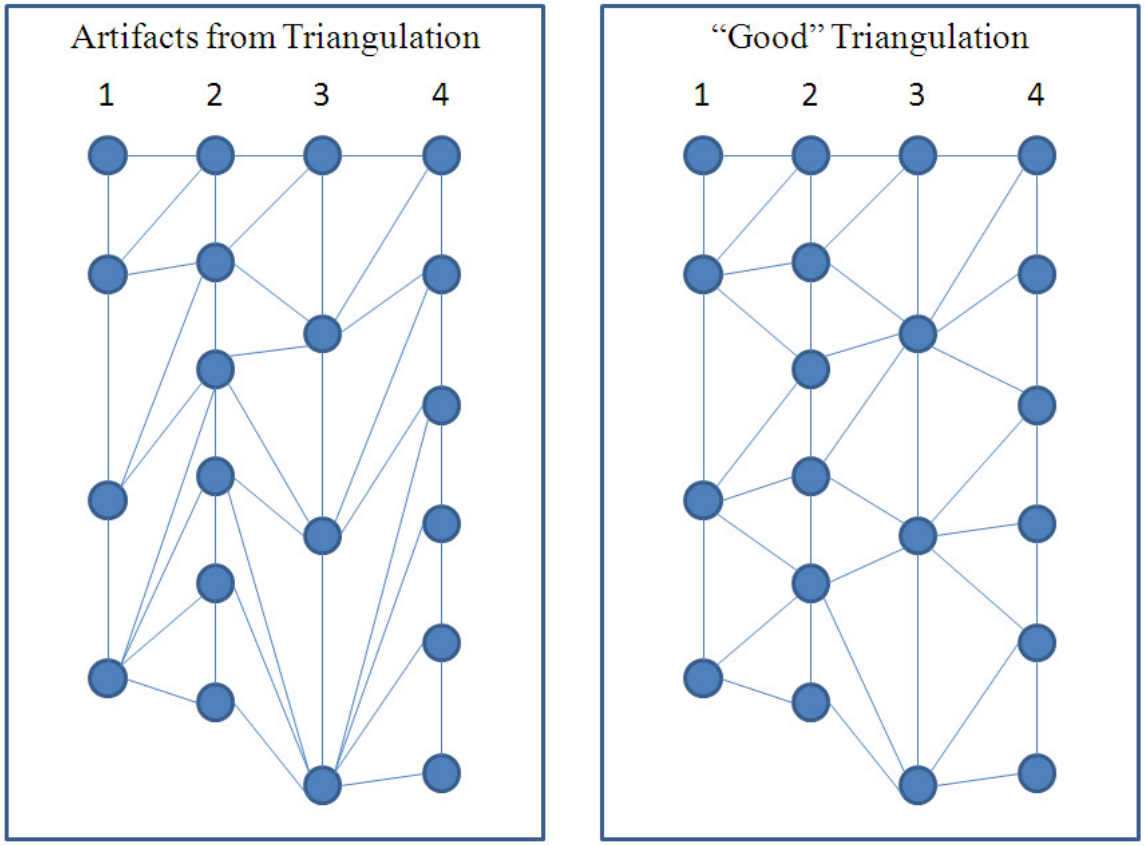


Figure 4.9 Possible artifacts from triangulation method (left) and a "good" triangulation (right). Note the left triangulation has triangles with small angles.

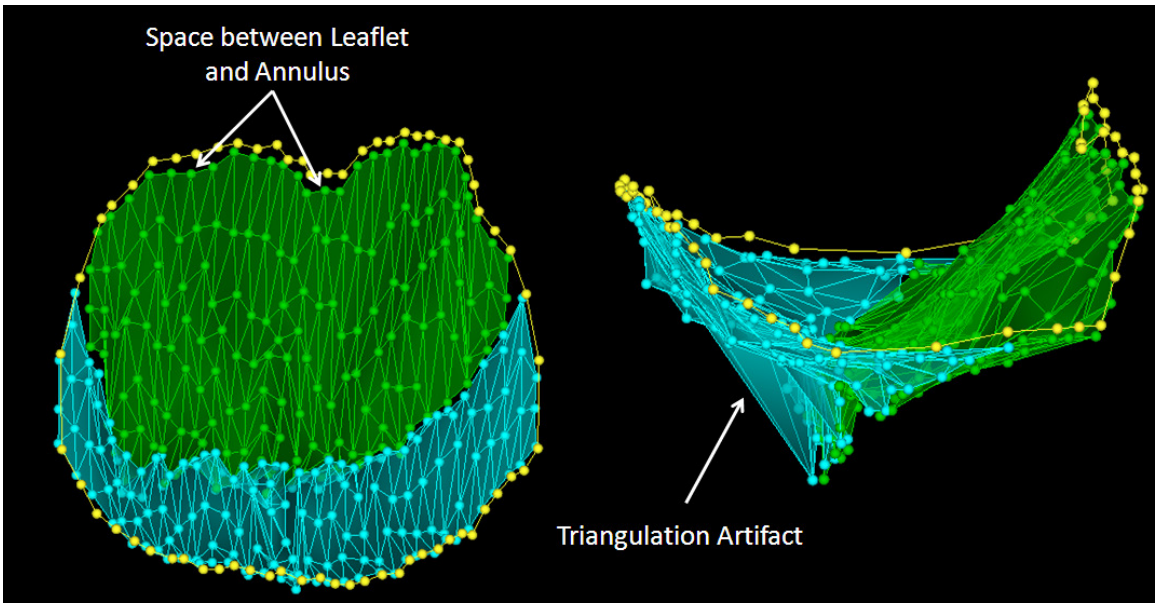


Figure 4.10 Drawbacks of triangulation with line segmentation method (yellow = annulus, green = anterior leaflet, blue = posterior leaflet). Connectivity between the annulus and leaflets is not defined and triangulation artifacts are present from variable number of points in neighboring segmentation images.

In an effort to ensure connectivity between the leaflets and annulus and reduce the number of user selected points required for segmentation, a segmentation method using user-selected curves was investigated. A curves-based method could also help eliminate triangulation artifacts because the curves could be sampled so that each neighboring data set contained the same number of points.

4.1.5 Curve Segmentation Method

The use of curves was investigated to improve upon the line segmentation method in three ways: ensuring connectivity between the leaflets and annulus, eliminating triangulation artifacts, and reducing the number of user-selected points required for

segmentation. The lack of connectivity between the leaflets and the annulus was addressed by defining the first point in the user generated curve as the nearest segmented annulus point. The triangulation artifacts were addressed by uniformly sampling each segmentation curve so that every neighboring slice contained the same number of points. The use of curves also reduced the number of user-selected points required because a few points could be used to define a shape instead of just a set of straight lines. Three different types of curves were investigated, including the *spline* function in MATLAB, Bezier curves and J-splines.

4.1.5.1 MATLAB Splines

Splines are curves defined in a piecewise manner by polynomial functions. In MATLAB, the *spline* function uses cubic polynomial functions to create a spline. The *spline* function defines a spline for a given set of points. The resulting spline could then be plotted on the segmentation image. While the user could generate a curve based upon their selected points, the *spline* function exhibited limitations that resulted in undesired curves. The *spline* function orders a set of given points in ascending order based upon their x-coordinate. It is assumed that the curve generated should pass through the points in this order instead of the order in which the points were given. This function also does not allow for multiple points to have the same x-coordinate. This could occur during segmentation depending on how the valve was oriented and leaflet geometry. For instance, when the leaflets are open during diastole, they could be aligned so that multiple points with the same x-coordinate are required to segment the leaflet. These two drawbacks resulted in the *spline* function not being pursued further. Instead, Bezier and J-spline curves were investigated because they are based upon sets of ordered points. This meant that the curves were not dependent upon either the x or y-coordinates of the selected points, only the order in which they are selected.

4.1.5.2 Bezier Curves

Bezier curves were investigated because they are based upon an ordered set of points. Bezier curves are defined as a convex combination of a set of points called a control polygon. A quadratic Bezier curve (Figure 4.11), which has a control polygon of four points (P_1,P_2,P_3,P_4), is defined parametrically by Equation 4.1. Where, t is evenly distributed from 0 to 1 in order draw the curve.

$$B(t) = (1-t)^3 P_1 + 3(1-t)^2 t P_2 + 3(1-t)t^2 P_3 + t^3 P_4 \quad (4.1)$$

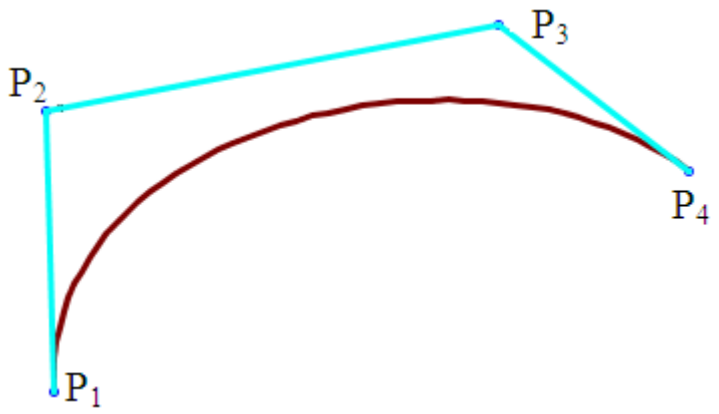


Figure 4.11 Example Bezier curve (red) with a control polygon (blue). Note: the Bezier curve does not pass through the points of the control polygon. (Modified screenshot taken from http://www.gvu.gatech.edu/~jarek/demos/retrofitBezier/).

Figure 4.11 shows that a Bezier curve generated by Equation 4.1 does not pass through the points in the control polygon (P_1,P_2,P_3,P_4). This would result in a user having to adjust the points of the control polygon away from segmentation area. Since it was desired to have the curve pass through the user-selected points, a retrofit scheme was needed. A retrofit scheme would ensure the Bezier curve would pass through each point

in the control polygon. Given four user-selected points P_1,P_2',P_3',P_4, the coordinates for the Bezier curve control polygon, (P_1,P_2,P_3,P_4) would be to be defined as shown in Figure 4.12. This was accomplished by constraining the values of the Bezier curve as follows:

$B(0) = P_1$

$B(1/3) = P_2$'

$B(2/3) = P_3$'

$B(1) = P_4$

solving the system of equations yields:

$$P_2 = \frac{-15P_1 + 54P_2' - 27P_3' + 6P_4}{18} \tag{4.2}$$

$$P_3 = \frac{-15P_4 + 54P_3' - 27P_2' + 6P_1}{18} \tag{4.3}$$

where:

P_1: First user-selected point and first point in the control polygon

P_4: Fourth user-selected point and fourth point in the control polygon

P_2: Second point in the control polygon

P_3: Third point in the control polygon

P_2': Second user-selected point

P_3': Third user-selected point

Equations 4.2 and 4.3 were used to generate the retrofitted Bezier curve depicted in Figure 4.12[30].

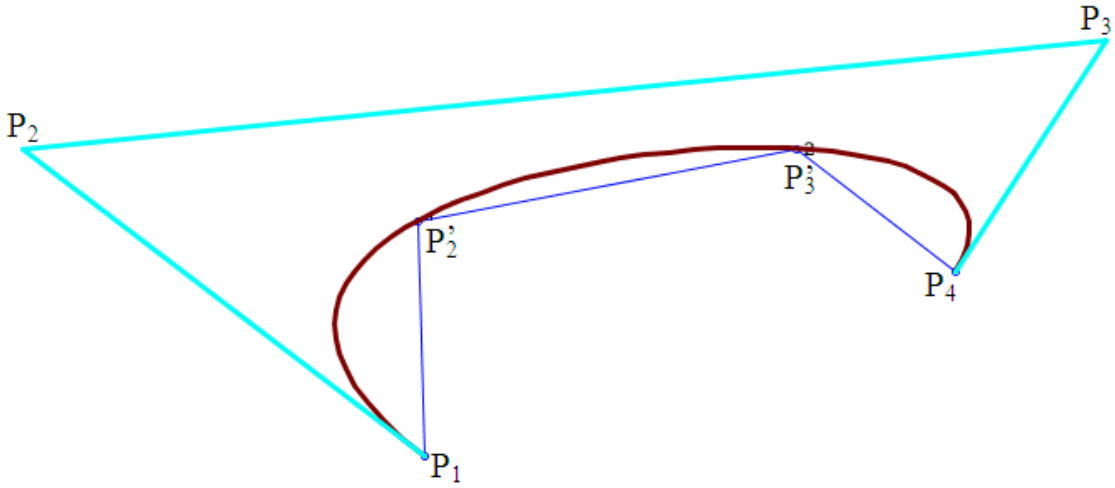


Figure 4.12 Retrofitted Bezier curve (red), original points (blue), and retrofitted control polygon (cyan) (Modified screenshot taken from http://www.gvu.gatech.edu/~jarek/demos/retrofitBezier/).

Since the user may require more or less than four points to segment a leaflet, a generalized equation for Bezier curves is required and defined by

$$B_n(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \tag{4.4}$$

where $B_n(t)$ is the Bezier curve defined parametrically for $0 \leq t \leq 1$ and P_i is the ith point in the control polygon.

The retrofit formulae for each number of points in the control polygon must be calculated in order for the generalized equation to be used. This can done by constraining

points, as in the case for the quadratic Bezier curve, and solving the system of equations. However, these formulae are different depending on the number of selected points. This would limit the maximum number of user-selected points and would be inefficient to program. Another drawback of using Bezier curves is that because they are a convex combination of each point within the control polygon, the curve generated is a global fit of all the points selected. In certain cases however, global fitting can be undesirable. For example, any added point to a set of existing selected points would alter the entire existing curve, possibly in a manner not desired by the user. With this in mind, local fitting was preferred for this application because it would allow the user to segment different parts of the leaflet without affecting other sections of the curve outside the current area of interest.

4.1.5.3 J-Splines

J-splines have desirable characteristics for segmenting mitral valve leaflets, including local fitting and ease of programming. J-splines are a set of subdivision curves based upon a generalized form of an arbitrary affine combination of the cubic B-spline and 4-point subdivision rule. A general J-spline, $J_{a,b}$ is defined by the Equations 4.5 and 4.6 below[31],

$$ {}^{k+1}P_{2j} = \frac{a^k P_{j-1} + (8-2a)^k P_j + a^k P_{j+1}}{8} \tag{4.5} $$

$$ {}^{k+1}P_{2j+1} = \frac{(b-1)^k P_{j-1} + (9-b)^k P_j + (9-b)^k P_{j+1} + (b-1)^k P_{j+2}}{8} \tag{4.6} $$

where ${}^kP_j$ represents the j^{th} control point at the k^{th} level of subdivision. If a is set equal to b, then these formulae define a subclass of J-splines, $J_{s,s}$, which will be denoted as J_s.

These encompass some known subdivision schemes, such as the four-point subdivision scheme (J_0) and the uniform cubic B-spline scheme (J_1)[31]. Figure 4.13 shows examples of J_0 (outer), $J_{1/2}$ (middle), and J_1 (inner) subdivision curves after 1,2, and 6 refinement steps. It can be seen that the J_0 curve contains the original vertices while the $J_{1/2}$ and J_1 curves do not.

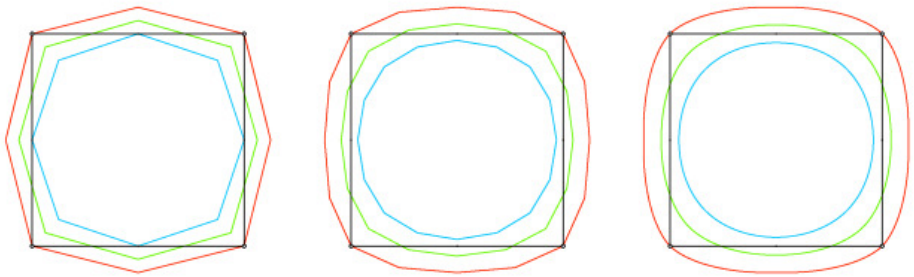


Figure 4.13 J-spline Subdivision curves after 1,2, and 6 refinements of J_0 (outer), $J_{1/2}$ (middle), J_1 (inner). Note the J_0 curves contain the original vertices[31].

It can be seen in Figure 4.13 that the outer curve J_0 interpolates the points of the square, while the inner curve J_1 and middle curve $J_{1/2}$ do not. It is important that the curve method used contains the user-selected points so that the user generates the curve by selecting points in the area of interest. If the curve did not contain the original points, the user would have to select points outside the leaflet to generate a curve on the leaflet. While retrofitting formulas can be applied so the J_1 and $J_{1/2}$ splines interpolate the points of the control polygon[31], J_0 splines were investigated because they interpolate a control polygon without the need for retrofitting formulas.

A *split and tweak* approach[32] was used to implement J-splines in MATLAB. This involved adding new vertices at the midpoints of all points in the current control polygon and then *tweaking* or moving the new points to fall on the J-spline. For implementation into MATLAB, the *tweak* method used a *tuck and untuck* approach[30].

The *tuck and untuck* method *tucks* a vertex by moving it a factor of 1/2 toward the midpoint of its neighbors and then u*ntucks* that vertex by moving it by a factor *s*-1 towards the midpoint of its neighbors. A *tuck* is defined as moving a vertex by a factor *s* of the vector from a vertex to the midpoint of its neighbors. An example of this is shown in Figure 4.14 where vertex *B* undergoes a *tuck* operation and *A* and *C* are its neighboring points, which influence the final position of vertex *B*. The vector *M* is defined as the vector from *B* to the midpoint of *A* and *C*. The new point *B'* is defined in Equation 4.7 where *s* is an input to the *tuck* operation and is the factor by which *M* is scaled.

$$B' = B + sM \tag{4.7}$$

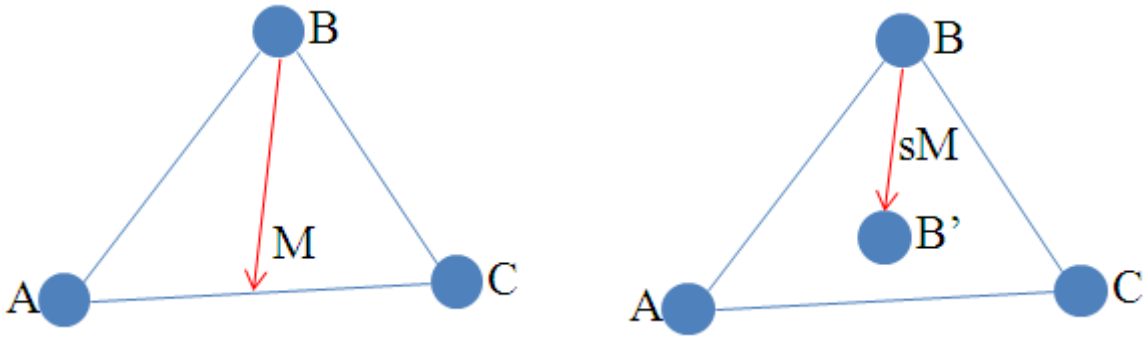


Figure 4.14 Tuck method where B is being *tucked* along M by a factor s.

For J-splines, the *tuck and untuck* method can be applied to create a J-spline of type J_s. This was accomplished by first refining the control polygon by inserting new points at the midpoints of all points in the current control polygon. After this, a *tuck*(1/2) followed by an *untuck*, defined as *tuck*(*s*-1), was applied to the refined control polygon. This method was applied to each point in the control polygon selected by the user to generate a refinement of the J-spline. J_0 spline refinement steps were made by applying *tuck*(1/2) and then *tuck*(-1). An example of this process is shown in Figure 4.15.

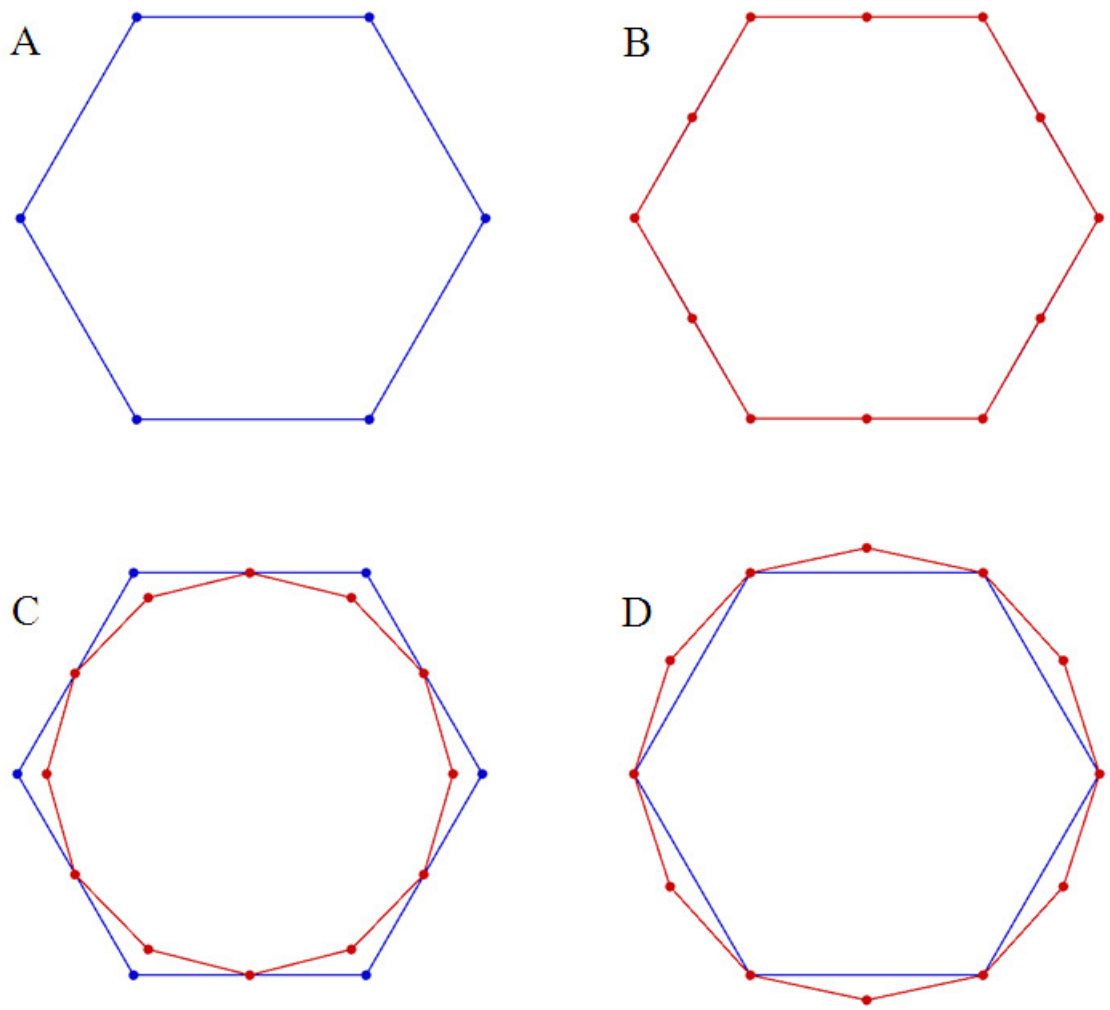


Figure 4.15 Example of tuck/untuck process to create a J_0 spline. (A) Original control polygon (blue). (B) Refined control polygon (red). (C) Refined polygon after "*tuck*" operation [*tuck*(1/2)] was applied. (D) Refined polygon after "*untuck*" operation [*tuck*(-1)] was applied. (Modified screenshots from http://www.gvu.gatech.edu/~jarek/demos/refine/).

To ensure that the J-spline passes through the end points of an open control polygon, two additional points were temporarily added to each end of an control polygon. An example of how these additional temporary points were defined and added is shown in Figure 4.16, while the additional points are defined in Equations 4.8 and 4.9. The first point added (*P1'*) was generated by calculating the vector (*AB*) between the endpoint of

the control polygon (*A*) and its nearest neighbor (*B*) and then subtracting that vector from the end point (*A*) (Equation 4.8).

$$P1' = A - AB \tag{4.8}$$

The second point (*P2'*) was generated by calculating the vector (*BC*) between the nearest neighbor (*B*) and the second nearest neighbor (*C*) to the endpoint and then subtracting that vector from the first point added (Equation 4.9).

$$P2' = P1' - BC \tag{4.9}$$

These additional points are depicted in Figure 4.16.

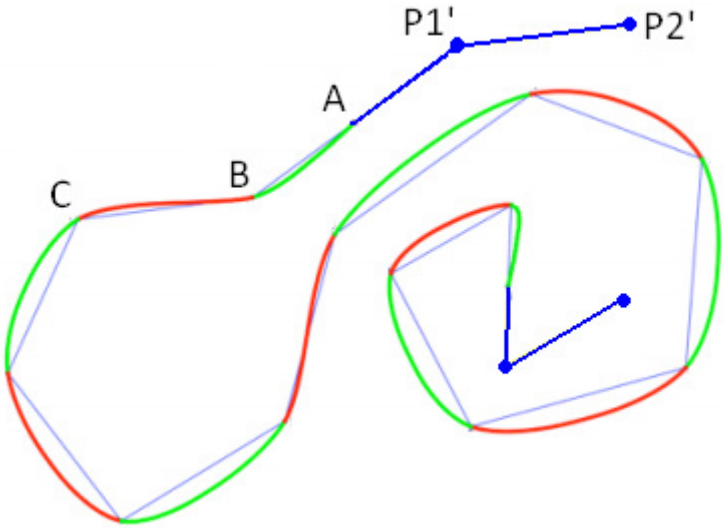


Figure 4.16 Additional endpoints P1' and P2' (blue) for the J-spline control polygon to ensure the spline passed through the original endpoint of the open control polygon [33].

where:

A: endpoint of the original control polygon

B: nearest neighbor to A in the original control polygon

C: nearest neighbor to B that is not A in the original control polygon

P1': first point added to the control polygon

P2': second point added to the control polygon

The final implementation of refining the J_0 was as follows:

1) Create two additional points on the end of each side of the control polygon based upon Equations 4.8 and 4.9
2) Create new vertices at the midpoints of every segment of the control polygon
3) Apply *tuck*(1/2) to the control polygon
4) Apply *tuck*(-1) to the control polygon
5) Remove temporary points beyond the original end points in the control polygon

4.1.5.4 Final Curve Selection

The final method for segmenting the leaflets used a J_0 spline generated by user selected points from the annulus to the free edge of the anterior and posterior leaflets. For display purposes, three refinement steps of the user-selected points were made to display a smooth curve. To eliminate any gap between the annulus and the leaflets the first point of each leaflet control polygon was defined as the nearest annulus point (Figure 4.17).

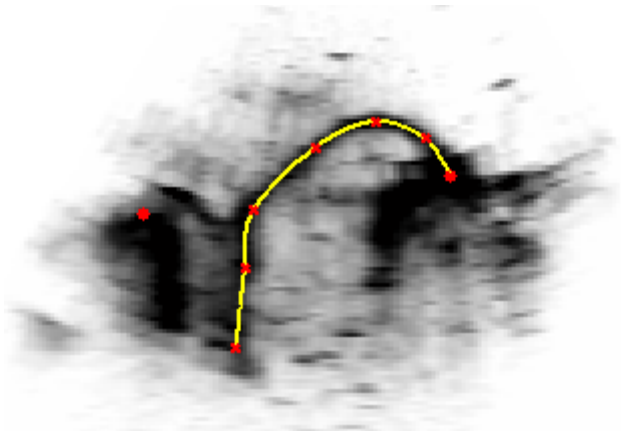


Figure 4.17 Final leaflet selection method (red = user-selected points, yellow = J-spline generated).

The following is a MATLAB script of how the segmentation occurs for the anterior leaflet:

```
i=0; % initialize index variable i (number of points selected)
button=1; % define button as pressed to enter while loop
while button==1
    [x,y,button]=ginput(1); % x,y and button number from button press
    x=round(x); % round for actual pixel value
    y=round(y); % round for actual pixel value
    if button==1
        if i==0 % find the nearest annulus pt and add to selected pts
            i=i+1; % increase the number of selected points by 1
            axData=meshData(s).annulus(:,1); % x of annulus points
            ayData=meshData(s).annulus(:,2); % y of annulus points
            [v,index]=min((axData-x).^2+(ayData-y).^2); % nearest pt
            ax=axData(index); % x value of nearest annulus point
            ay=ayData(index); % y value of nearest annulus point
            meshData(s).anterior(i,:)=([s ay ax]); % store point
        end
        i=i+1; % increase the number of selected points by 1
        meshData(s).anterior(i,:)=([s y x]); % store selected point
        px=meshData(s).anterior(:,3); % define x values to be plotted
        py=meshData(s).anterior(:,2); % define y values to be plotted
        finalPts = tweak4pt([px' py']); % define x,y for J-spline
        for j = 1:1:3 % subdivide J-spline 3 times
            finalPts = tweak4pt(finalPts);
        end
        xx = finalPts(:,1); % x values of subdivided J-spline
        yy = finalPts(:,2); % y values of subdivided J-spline
        plot(xx,yy,'y','LineWidth',2)   % plot yellow J-spline
        plot(px,py,'rx','LineWidth',2); % plot red Xs of selected pts
```

```
    end
end
```

where:

i: number of points selected

button: button number last pressed

x: x-coordinate of the most recently selected point

y: y-coordinate of the most recently selected point

s: slice number of the image being segmented

meshData.anterior: storage variable for the selected points in a particular image

px: x-coordinates of all selected points

py: y-coordinates of all selected points

axData: x-coordinates of the selected annulus points

ayData: y-coordinates of the selected annulus points

index: index of the closest annulus point in axData and ayData

tweak4pt: function to subdivide the control polygon based upon a J_0 spline

finalPts: variable to store the x and y-coordinates of the J-spline control polygon

xx: x-coordinates of the subdivided J-spline

yy: y-coordinates of the subdivided J-spline

Once the first point is selected, the nearest annulus point for the current frame is found and added as the first point in *meshData.anterior*. The first selected point is added as the second point in *meshData.anterior*. Then, *finalPts* is used to store the subdivided J-spline while *tweak4pt* is used to subdivide the selected points 4 times to generate the J-spline to be plotted. Any additional selected points are added to *meshData.anterior* and the J-spline is updated and plotted.

Once the entire valve was segmented the curves were subdivided until the distance between two neighboring points in the control polygon was less than 1% of the total length of the curve, which was approximated by the sum of the distances between the points in the subdivision curve. The subdivided curve was then sampled for a number of equidistant points along the curve from the annulus to the free edge of the leaflet. This meant that before triangulation, each segmented image contained the same number of points for each leaflet. Therefore, the artifacts induced by differences in the number of points defining each leaflet within each segmented image were eliminated.

4.1.6 Graphical User Interface Development

One consideration for usability of the program was the Graphical User Interface (GUI). The objective of the GUI is to provide a graphical interface for the user to interact with the segmentation tool and view any information necessary to complete the segmentation. If a GUI does not provide enough information or is unorganized it can be difficult to manually segment the valve. If the GUI has too much information on it and is cluttered, it can also make the tool harder to use by distracting the user from the main task. An optimal GUI would provide the user with the necessary information to complete the segmentation and have a logical layout to guide the user through the segmentation process.

4.1.6.1 Single View GUI

During the initial development of the segmentation tool using the point segmentation, a single stack of images was used to segment the valve. This GUI provided the user with a single view of the mitral valve being segmented as seen in Figure 4.18.

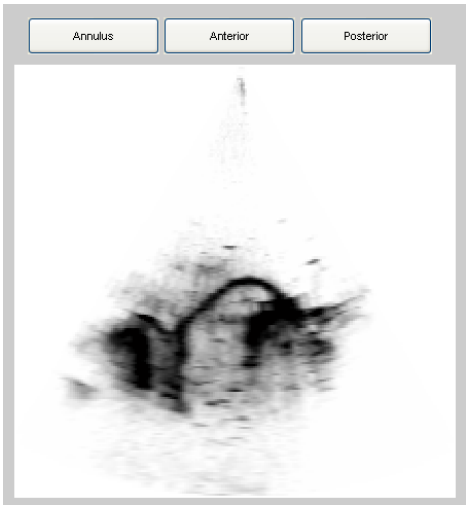


Figure 4.18 Single view GUI displaying a closed mitral valve.

This GUI only provides the 2D segmentation planes for a single time point. Other information that could help improve the segmentation includes: viewing other time points, an annulus plane view, and a plot of the currently segmented points. To provide the user with this additional information, a quad view GUI was developed in conjunction with the line segmentation method to provide the user with additional information to aid in segmentation.

4.1.6.2 Quad View GUI

The quad view GUI contained four different views. The top-left view contained the image to be segmented. The top-right view contained a 2D view of the annulus plane. The bottom-left image was a commissure-to-commissure plane of the valve. The bottom-right image contained all of the currently segmented points plotted in three-dimensional space. A screenshot of this GUI is shown in Figure 4.19.

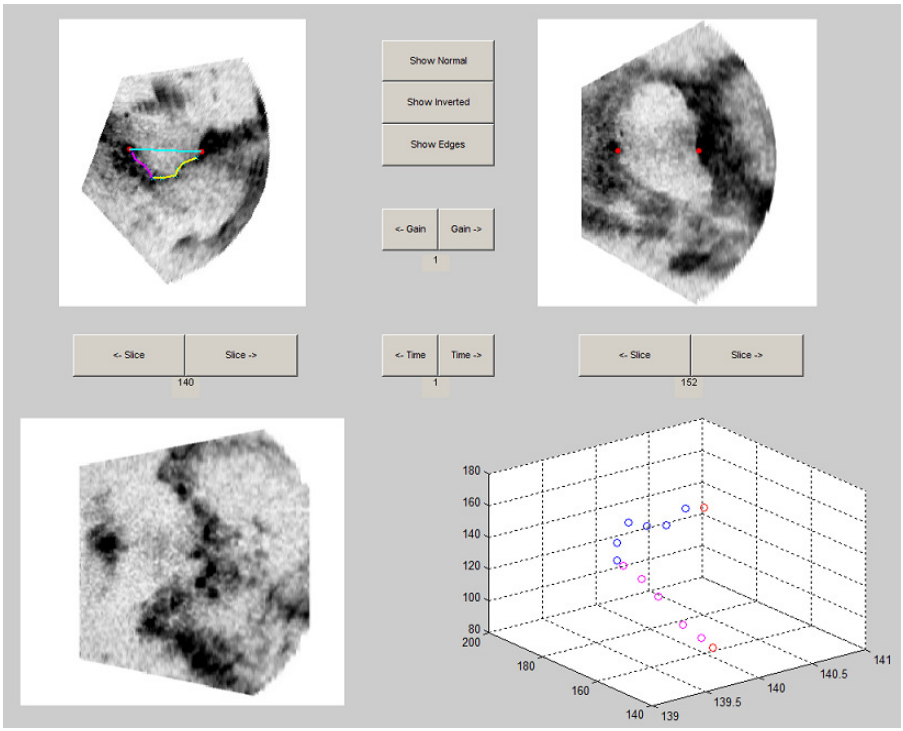


Figure 4.19 Quad view GUI with the segmentation view (top-left), annulus view (top-right), commissure-commissure slice (bottom-left) and 3D plot of segmented points (bottom-right).

The quad view GUI accomplished the goal of providing the user with more information about the segmentation. However, the lower views did not provide information that would improve the segmentation. The two most important views for

segmentation information were the anterior-posterior view of the leaflets and the annulus plane. In light of this, a dual view GUI was investigated.

4.1.6.3 Dual View GUI

The dual view GUI was developed to magnify the segmented image displayed, and only provide the images that the user needs. The dual view GUI has the main segmentation window on the left and the supplemental view of the mitral annulus on the right as displayed in Figure 4.20.

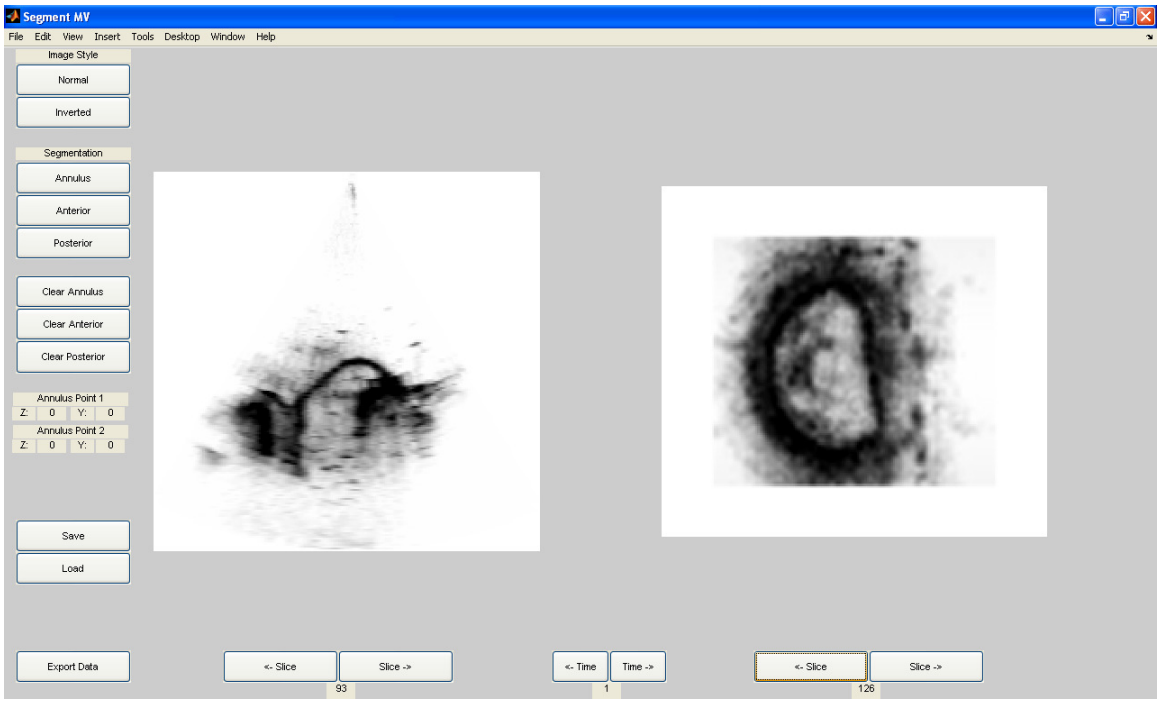


Figure 4.20 Final (dual view) GUI for the segmentation program containing the segmentation view on the left and the annulus view on the right.

This GUI provided the user with the main segmentation view and an atrial side view of the mitral annulus, which provided the main information necessary to perform the segmentation.

4.1.7 DICOM Scaling

Due to variability in the location of the mitral valve in echocardiography data sets, a tool was developed to scale, rotate, and crop Cartesians DICOMs to provide the user with a consistent orientation to segment the mitral valve within the segmentation tool.

The Cartesian DICOM obtained using the Philips® QLAB software resulted in the length scales between each of the *x*, *y*, and *z*-directions of the Cartesian DICOM being unequal (non-cubic voxels). Since the data set may need to be rotated to obtain the desired orientation for segmentation, the differences in length scale in each direction would need to be taken into account. In order to more easily rotate the data, a tool to scale the Cartesian DICOM to create cubic voxels with sides 0.5mm in length was implemented.

In general, the Cartesian DICOM files had a directional spacing of 0.4-0.5mm in each direction, and a spacing of 0.5mm for each direction was chosen to create cubic voxels. To accomplish this, the Matlab function *imresize* was used. This function scaled a 2D image file of size *x* by *y*, to a given size *x'* by *y'*. The tool calculated the closest number of pixels required for a pixel spacing of 0.5mm in each direction. The tool then used nested *for* loops to scale each 2D image in the entire 4D data set. Because the number of pixels must be an integer, there was a rounding error associated with this scaling, however this was always less than 1%.

The following MATLAB code depicts the Cartesian DICOM scale method:

```
dicom.data % 4D dicom data
[xSize,ySize,zSize,tSize]=size(dicom.data); % get size of dicom
mmScale=0.5; % desired mm/pixel value
xCurScale=dicom.mmPerPixelX; % current mm/pixel for x-direction
yCurScale=dicom.mmPerPixelY; % current mm/pixel for y-direction
zCurScale=dicom.mmPerPixelZ; % current mm/pixel for z-direction

% calculate scale percentage for each direction
xScale = xCurScale/mmScale;
yScale = yCurScale/mmScale;
zScale = zCurScale/mmScale;

% calculate closest number of pixels to 0.5mm/pixel
xNumNew = round(xNum*xScale);
yNumNew = round(yNum*yScale);
zNumNew = round(zNum*zScale);

% loop to scale images in y and z directions
for i = 1:1:tSize
    for j = 1:1:xSize
        img = squeeze(dicom.data(j,:,:,i));
        imgScaled = imresize(img,[yNumNew zNumNew]);
        dataout(j,:,:) = imgScaled;
    end
    dataFinal(:,:,:,i) = dataout;
end
dataFinal = permute(dataFinal,[3 2 1 4]);
dicom.data = dataFinal;
dataFinal = []; % clear dataFinal to save memory

% loop to scale images in x direction
for i = 1:1:tSize
    for j = 1:1:zNumNew
        img = squeeze(dicom.data(j,:,:,i));
        imgScaled = imresize(img,[yNumNew xNumNew]);
        dataout2(j,:,:) = imgScaled;
    end
    dataFinal2(:,:,:,i) = dataout2;
end
dicom.data = []; % clear reduce memory usage
dataFinal2 = permute(dataFinal2,[3 2 1 4]);
dicom.data = dataFinal2;
```

where:

dicom.data: 4D DICOM data set

xSize,ySize,zSize,tSize: size of the x,y,z,t in the DICOM data set

mmScale: desired mm/pixel value

xCurScale, yCurScale, zCurScale: current mm/pixel for each direction

xScale, yScale, zScale: percent change in pixel length required to match mmScale

xNewNum, yNewNum, zNewNum: pixel length for each direction required to match mmScale

img: single 2D image from the 4D DICOM data set

squeeze: MATLAB function to remove singleton dimensions from a matrix

imgScaled: scaled 2D image

imresize: MATLAB function to resize a 2D image to the new number of pixels passed to it

dataout: storage for single timepoint of 3D rotated data

dataFinal: storage for rotated 4D data set

The *imresize* function can only resize 2D images, so two *for* loops were implemented in order to scale the *y* and *z*-directions in the first loop and the *x*-direction in the second.

4.1.8 DICOM Rotation

A DICOM rotation program was implemented in MATLAB to orient the mitral valve for segmentation. The DICOM was oriented such that the top-left (segmented) image was an anterior- posterior view of both leaflets, and the top-right image contained the annulus plane (Figure 4.21).

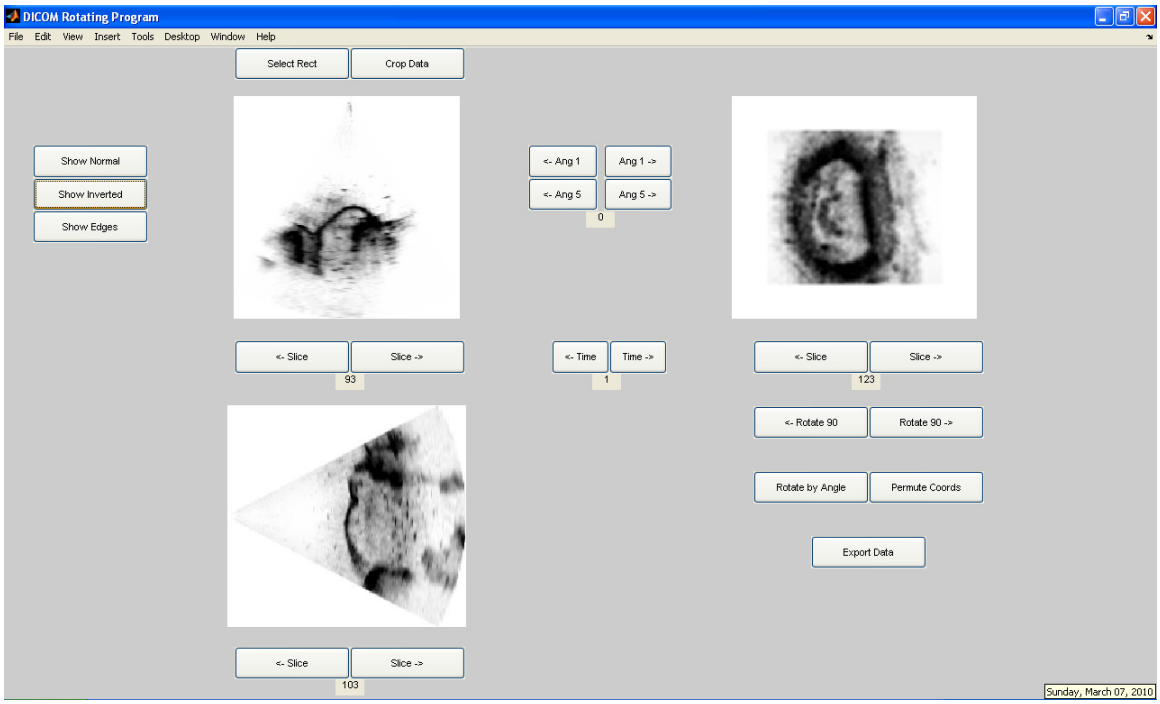


Figure 4.21 DICOM rotate and crop GUI depicting the segmentation view in the top-left and the atrial view of the annulus in the top-right image.

The GUI (Figure 4.21) displayed the *xy*, *yz*, and *xz*-planes of the Cartesian DICOM at a single time point. To orient the data set, the user would select "Permute Coords", which would change the position of the *x*, *y*, and *z*-coordinates using the MATLAB function *permute*. The user permuted the data set until annular plane was displayed in the top-right image and the anterior-posterior view was displayed in the top-left image. After this, the user rotated the top-left image using the angle buttons until left atrium was located at the top of the image, the left ventricle at the bottom of the image, and the annulus was approximately horizontal in the image. The angle buttons rotated the top-left image in clockwise and counter-clockwise directions in one or five degree increments using the *imrotate* MATLAB function. Once the desired rotation was

reached, the entire data set was rotated using the 2D *imrotate* function and nested *for* loops as shown in the following MATLAB code:

```
rAngle=5; % angle of rotation
dicom.data; % 4D dicom data
[xSize,ySize,zSize,tSize] = size(dicom.data); % get size of dicom

% rotate each YZ plane and assemble back into dataFinal
for i=1:1:tSize
    for j=1:1:xSize
        img=squeeze(dicom.data(j,:,:,i)); % create 2D image
        imgrotated=imrotate(img,rAngle,'crop'); % rotate by rAngle
        dataout(j,:,:)=imgrotated; % store rotated 2D image
    end
    dataFinal(:,:,:,i)=dataout; % store rotated timepoint
end
dicom.data=dataFinal; % update 4D dicom
```

where:

rAngle: angle of rotation in degrees

dicom.data: 4D DICOM data set

xSize,ySize,zSize,tSize: size of the x,y,z,t in the DICOM data set

img: single 2D image from the 4D DICOM data set

squeeze: MATLAB function to remove singleton dimensions from a matrix

imgrotated: rotated 2D image

imrotate: MATLAB function to rotate a 2D image by an angle in degrees

dataout: storage for single timepoint of 3D rotated data

dataFinal: storage for rotated 4D data set

4.1.9 DICOM Cropping

To reduce the size of the data set and improve program speed, a cropping method was implemented within the DICOM rotation program. This would eliminate parts of the data set that were not required for segmentation. Once the user selected the area of interest for segmentation, the entire DICOM was cropped using the MATLAB function *imcrop* and nested *for* loops as shown in Figure 4.22.

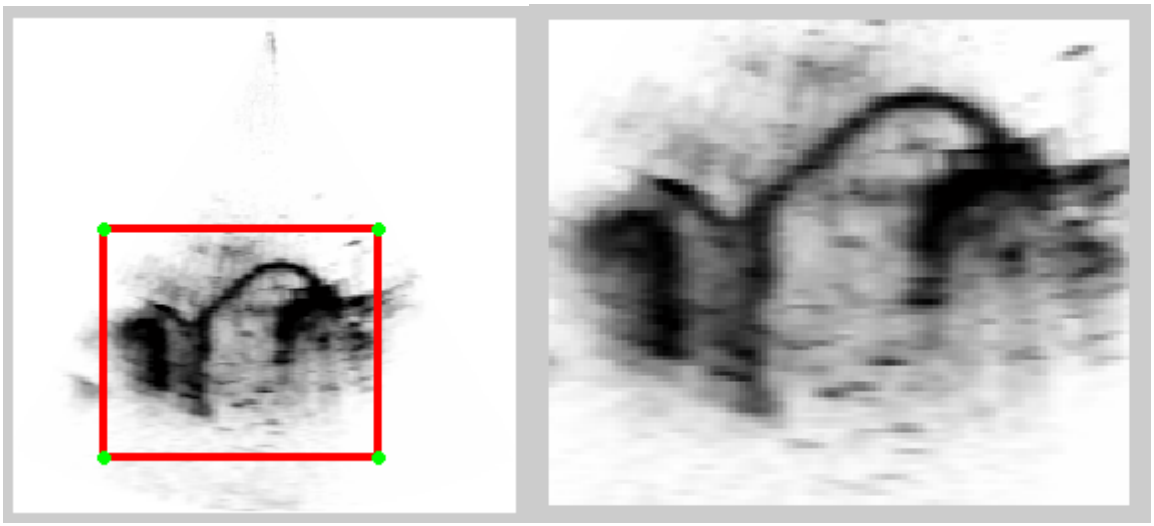


Figure 4.22 DICOM crop selection (left) and cropped image (right).

The following MATLAB code shows how the cropping code was implemented:

```
dicom.data; % 4D dicom data
[xSize,ySize,zSize,tSize] = size(dicom.data); % get size of dicom
cropVal = [xmin ymin width height]; % rectangle to crop with
for i = 1:1:tSize
    for j = 1:1:xSize
        img = squeeze(dicom.data(j,:,:,i)); % create 2D image
        imgCropped = imcrop(img,cropVal); % crop with cropVal
        dataout(j,:,:) = imgCropped; % store rotated 2D image
    end
    dataFinal(:,:,:,i) = dataout; % store cropped timepoint
end
dicom.data=dataFinal; % update 4D dicom
```

where:

dicom.data: 4D DICOM data set

xSize,ySize,zSize,tSize: size of the x,y,z,t in the DICOM data set

cropVal: array to define cropping rectangle

img: single 2D image from the 4D DICOM data set

squeeze: MATLAB function to remove singleton dimensions from a matrix

imgCropped: cropped 2D image

imcrop: MATLAB function to crop a 2D image by a rectangle

dataout: storage for single timepoint of 3D cropped data

dataFinal: storage for cropped 4D data set

4.1.10 Segmentation Protocols

In order to complete a segmentation of a data set, two separate protocols were used for first preparing the data set and then segmenting the data set.

The protocol for preparing the data set is as follows:

1) Export standard DICOM from QLAB as Cartesian DICOM.
2) Read Cartesian DICOM into MATLAB.
3) Scale Cartesian DICOM using *scaleDicom* function.
4) Import scaled Cartesian DICOM into DICOM rotation program, *rotateMV*.

5) Rotate and permute 4D data set until the correct segmentation view is obtained on the top-left image and an annular view is obtained on the top-right image.

6) Use the cropping tool to select the correct area to crop the DICOM to.

7) Save the updated DICOM.

The protocol for the segmentation program is as follows:

1) Load DICOM into *segmentMV* program.

2) Select the instance in time desired to segment.

3) Adjust the top-right image until an annular view is obtained

4) Using the segmentation buttons, select the two annulus points in the image.

5) Select the anterior leaflet.

6) Select the posterior leaflet.

7) Repeat until each slice across the leaflet is segmented.

8) Once the segmentation is complete export the segmentation, generating the virtual models for the annulus, anterior leaflet, and posterior leaflet.

4.2 Mesh Refinement Methodologies

Three mesh refinement methodologies were investigated. The first two methods, Loop and Butterfly refinement, are well-known methods of refining a triangular mesh. The third method was developed for this specific application of segmenting mitral valves and interpolated between the segmented images based upon 4-point subdivision curves.

Using a fully sampled data set as the gold standard, each method was evaluated by interpolating a half-sampled data set and then comparing the result to the fully samples data set. This was done for the peak systolic cases for a normal valve, a valve with a flail leaflet, and a valve with a billowing leaflet.

4.2.1 Corner Table Representation

A corner table is a way to represent and traverse a triangle mesh[34]. This representation was used in investigating and developing the mesh refinement methodologies. Mesh information is stored in multiple tables to facilitate traversing and manipulating the mesh. The corner table representation consists of a G-Table, V-Table, O-Table, and Next & Previous-Tables. The G-Table stores the *x*, *y*, and *z*-coordinates of each vertex in the triangle mesh. The V-Table stores the index in the G-Table of the vertex for each corner in the triangle mesh. The O-Table stores the V-Table index of the opposite corner, where the opposite corner is defined as corners that share the same next and previous corners. In addition, a table with the V-Table index of the next and previous corners is also generated to complete the representation. Figure 4.23 provides a representation of the data in the corner table.

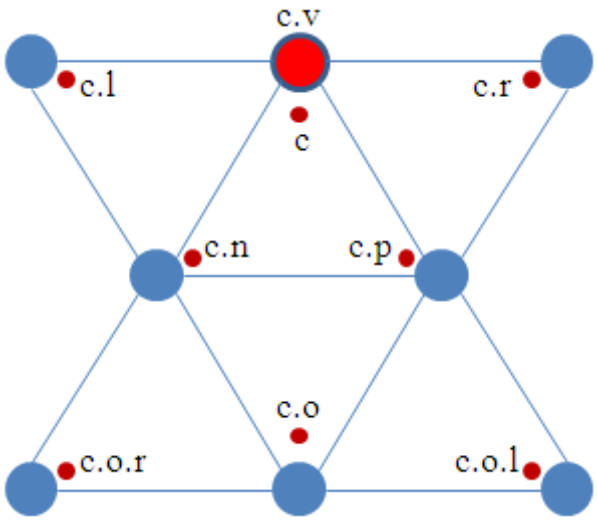


Figure 4.23 Example of corner table values.

where:

c: current corner

c.v: vertex of c

c.p: previous corner to c in the orientation of the triangle (counter-clockwise)

c.n: next corner to c in the orientation of the triangle (counter-clockwise)

c.o: corner opposite to c

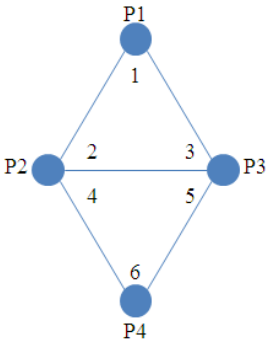
c.l: corner left of c

c.r: corner right of c

c.o.l: corner left of c.o

c.o.r: corner right of c.o

An example of the corner table values for a simple two triangle mesh is shown in Figure 4.24.



G-Table			V-Table	O-Table	N-Table	P-Table
x_{P1}	y_{P1}	z_{P1}	1	6	2	3
x_{P2}	y_{P2}	z_{P2}	2	-1	3	1
x_{P3}	y_{P3}	z_{P3}	3	-1	1	2
x_{P4}	y_{P4}	z_{P4}	2	-1	6	5
			3	-1	4	6
			4	1	5	4

Figure 4.24 Example of corner table values for a simple two triangle mesh.

Figure 4.24 shows the G-Table stores the *x*, *y*, and *z*-coordinates of points *P1*,*P2*,*P3*, and *P4* in their respective columns. The V-Table shows the vertex index for the corners 1 to 6. In the O-Table -1 represents a border edge that has no opposite corner. It can be seen in Figure 4.24 that only corners 1 and 6 have opposite corners, which are each other. The Next (N) and Previous (P) Tables store the index for the next and previous corners when traversing the triangle in a counter-clockwise direction. For example, the first values in the N and P tables represent the next and previous corners relative to corner 1, which are 2 and 3 respectively.

4.2.2 Generating a Corner Table from Face-Vertex Data

The corner table can be computed from a consistently oriented triangle mesh imported from a PLY file. When the PLY file is imported into MATLAB, the vertex and face data are placed into separate variables. Then the vertex data is defined as the G-

Table. Finally, the V, O, and Next/Previous Tables are generated by traversing the face data.

4.2.3 Loop Refinement

The first mesh refinement method investigated was Loop's subdivision method. Loop's refinement method generates four new triangles for every triangle in the original mesh and generates new vertices as a weighted combination of its surrounding points[35]. A new point, *P*, is generated according to the weighting scheme depicted in Figure 4.25 and defined in Equation 4.10.

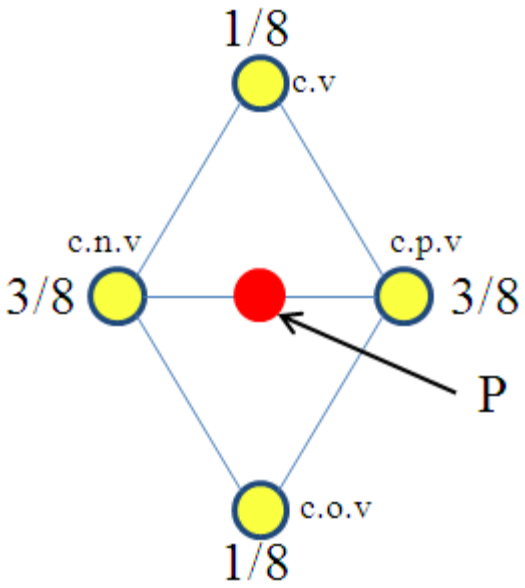


Figure 4.25 Loop's weighting scheme for creating new vertices (new vertex = P).

where:

c.v: vertex of corner c

c.p.v: vertex of the corner previous to c

c.n.v: vertex of the corner next to c

c.o.v: vertex of the corner opposite c

P is defined for non-border edges in Equation 4.10.

$$P = \frac{c.v + 3(c.p.v) + 3(c.n.v) + c.o.v}{8} \tag{4.10}$$

For border edges, *P* is defined as the midpoint between *c.n.v* and *c.p.v*. The original vertices were recalculated as a weighted average of 5/8 of the original coordinate and 3/8 of the average coordinate of its neighborhood. This scheme was implemented in MATLAB in the following steps:

1) Generate corner table from PLY triangular mesh file
2) Create new non-border points based upon Loop weighting scheme and new border points at the midpoint of *c.n.v* and *c.p.v*. Recalculate original points.
3) Define new triangles (4 in each original triangle)
4) Export new mesh to PLY triangular mesh file.

4.2.4 Butterfly Mask

Another mesh refinement method investigated for reducing the segmentation time was the butterfly mask[36]. This method also generates four triangles for every triangle in the original mesh and generates new points on each non-border edge. The butterfly mask is based upon a weighted average of eight surrounding points instead of four as pictured in Figure 4.26 and defined in Equation 4.11.

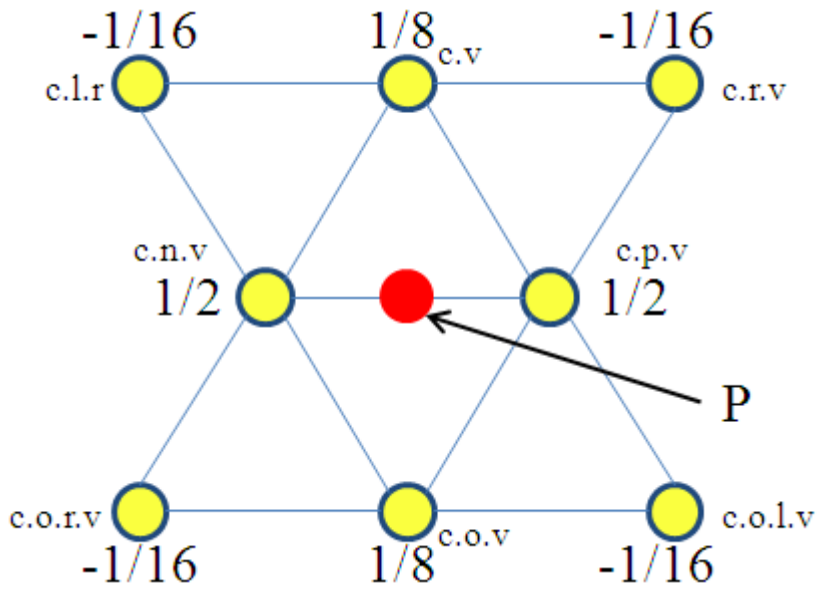


Figure 4.26 Butterfly weighting scheme for creating new points (new vertex = P).

where:

c.v: vertex of corner c

c.p.v: vertex of the corner previous to c

c.n.v: vertex of the corner next to c

c.o.v: vertex of the corner opposite c

c.l.v: vertex of the corner left of c

c.r.v: vertex of the corner right of c

c.o.l.v: vertex of the corner to the left of the corner opposite c

c.o.r.v: vertex of the corner to the right of the corner opposite c

P is defined for non-border edges in Equation 4.11.

$$P = \frac{2(c.v) + 8(c.p.v) + 8(c.n.v) + 2(c.o.v) - c.l.v - c.r.v - c.o.l.v - c.o.r.v}{16} \tag{4.11}$$

For border edges, P is defined as the midpoint between *c.n.v* and *c.p.v*. This scheme was implemented in MATLAB in the following steps:

1) Generate corner table from PLY triangular mesh file
2) Create new non-border points based upon Butterfly weighting scheme and new border points at the midpoint of *c.n.v* and *c.p.v*.
3) Define new triangles (4 in each original triangle)
4) Export new mesh to PLY triangular mesh file.

4.2.5 Inter-slice Interpolation

In addition to the Loop and Butterfly mesh refinement methods, another method was developed to interpolate additional data points across the valve based upon four-point subdivision. Each J_0 spline created for the leaflets was sampled to have the same number of points. Therefore, a J_0 spline can be created using the *i*th point in each segmentation slice as the control polygon (Figure 4.27). The curve is then subdivided until the distance between the points is less than 1% of the total length of the curve. The J_0 spline is then sampled at the new number of desired points across the valve. The new points are then triangulated using the same scheme described earlier in section 4.1.4.

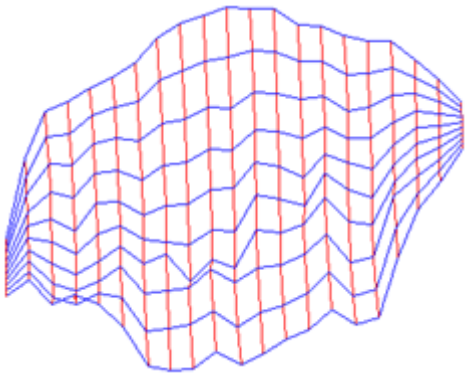


Figure 4.27 Example of J-spline (blue) created between the segementation slices (red) for an anterior leaflet.

4.3 Validation Methodology

Validation of the manual segmentation tool was accomplished using stereo photogrammetry on a dynamic, *in vitro* porcine mitral valve model. 3D tissue coordinates of tissue dye markers on the valve leaflets were obtained through stereoscopic reconstruction using a direct linear transformation. The 3D coordinates obtained from tissue dye markers were compared to the virtual model generated by the segmentation tool after a transformation to match their coordinate systems. The 3D marker data were also compared to the virtual model after a best fit alignment was applied.

4.3.1 Pulsatile *in vitro* left heart simulator

Dynamic experiments were performed in the Georgia Tech Left Heart Simulator[37-39]. The simulator consists of an acrylic left ventricular chamber, acrylic annulus plate, acrylic atrial chamber, stainless steel papillary muscle positioning rods, a resistance valve, a capacitance chamber, and an atrial reservoir. In addition to these

components, a bladder pump system controlled by a pulse duplicator was used to generate flow into the ventricular chamber. The bladder pump comprised of a silicone bladder (Clear Silicone Shore 30- Cured at 400 F for 2 hours, Hi-Tech Elastomers Co Ltd, CA) enclosed in an air-tight acrylic cylinder (Part # 8486K593, McMaster-Carr, GA) with 3 ports to allow filling (1 port) and evacuation (2 ports) of air from the cylinder. The ports were connected to a circuit of programmable solenoid valves (Model 6213, Burkert Fluid Control Systems, Irvine, CA) and an air compressor (Part # T-35HD-1 HP Ultra Oil-Less Air Compressor, Thomas Air-Pac, WI) that were controlled by an in-house DOS program (Trigger.exe). The working fluid was 0.9% saline. A diagram of the dynamic left heart simulator is depicted in Figure 4.28. This system has been previously used in multiple mitral valve studies to mimic physiologic conditions[37-39].

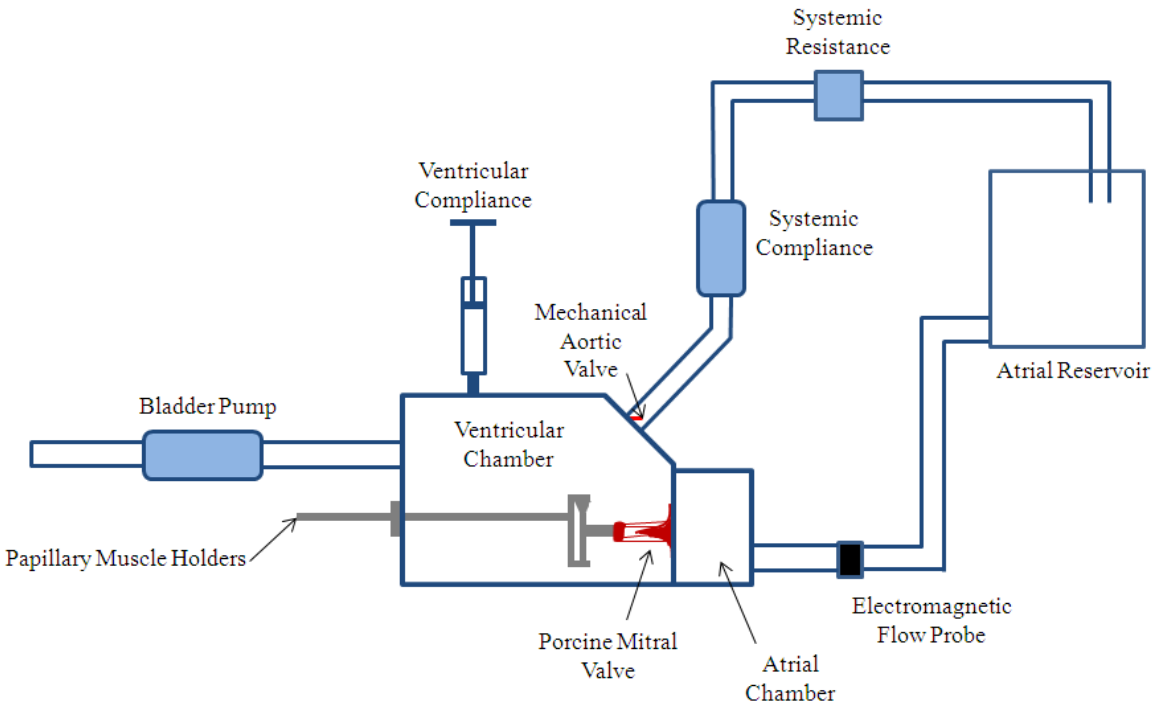


Figure 4.28 Diagram of the left heart simulator used for dynamic experiments.

For dynamic experiments, the left heart simulator was tuned to achieve 120 mmHg transmitral pressure and average cardiac output of 5 L/min. Pressure was measured in the atrium and ventricle using Edwards Lifesciences (Irvine, CA) TruWave pressure transducers (Model PX600). A Carolina Medical Electronics (East Bend, NC) electromagnetic flow meter (Model EP680 – 25.4mm diameter) was used to measure flow rates. A mechanical tilting disc aortic valve (Medtronic Hall Valve, MN) was used to prevent backflow to the ventricular chamber back from the outflow tract. A side inlet atrial chamber was designed and built for optical access for the high-speed cameras and to minimize the speed of sound and refraction errors of the echocardiography measurements (Figure 4.29).

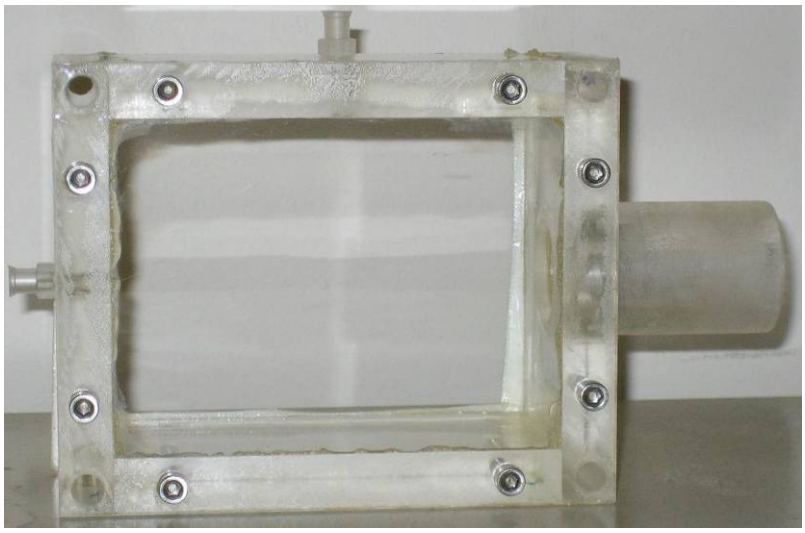


Figure 4.29 Side inlet atrial chamber.

Pressure and flow rate data were simultaneously monitored and recorded using a data acquisition system (DAQ). The DAQ system consisted of a BNC-2110 Isolated Signal Connector (National Instruments, Austin, TX), PCMCIA DAQCard-6024E (National Instruments, Austin, TX), and an in-house data acquisition and monitoring system (Heartbreaker) written in LABVIEW (National Instruments, Austin, TX).

4.3.2 Valve Selection and Preparation

Mitral valves were excised from fresh porcine hearts obtained from a local abattoir (Holifield Farms, Covington, GA). Diseased valves were excluded and preference was given to mitral valves possessing two distinct papillary muscles due to the restrictions in mounting the papillary muscles to the rods in the chamber. The entire mitral valve apparatus (annulus, leaflets, chordae, and papillary muscles) was preserved during excision and stored in 0.9% saline prior to each experiment. After excision, valves were sutured to a fixed size annulus plate. Both leaflets were marked with a grid of dots using tissue dye (Shandon dye, Thermo Scientific, Waltham, MA) with spacing of approximately 2-3mm as pictured in Figure 4.30.

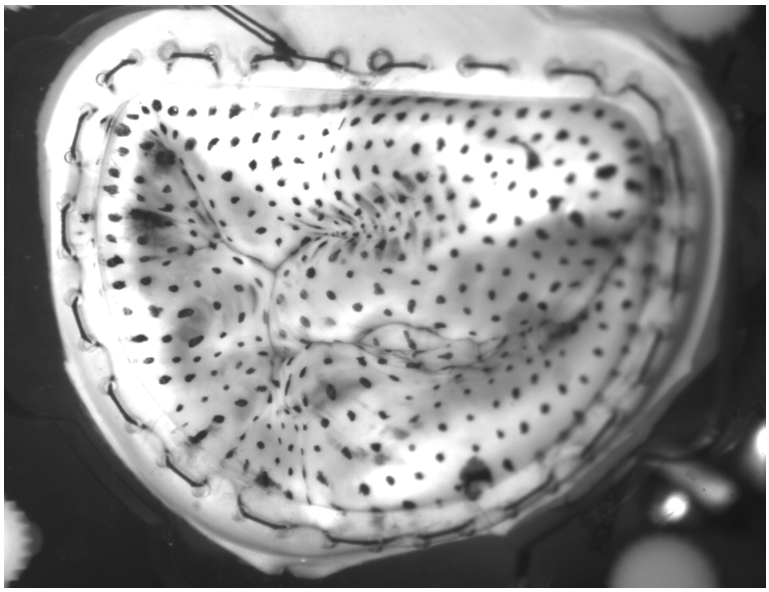


Figure 4.30 Mitral valve with tissue dye markers.

Three different states of the mitral valve were examined in this study. A normal valve state was created by adjusting the papillary muscles so that normal leaflet coaptation was

obtained. The flail leaflet disease state was created by lengthening the marginal and strut chordae of the anterior leaflet with sutures. The billowing disease state was created by lengthening only the strut chordae with sutures. Chordae lengthening was accomplished by placing a suture through each chordae with slack and then transecting the chordae. Papillary muscles were left in their normal position for both disease cases.

4.3.3 High-Speed Camera System

Two Basler 504k (Basler Corp, Germany) high resolution frame grabbers with Nikon 105mm f:1-2.8 AF Micro-Nikkor lenses (Nikon Inc., Melville, NY) were used to acquire simultaneous high speed images through an EPIX (EPIX Imaging, Buffalo Grove, IL) high-speed acquisition system. This system was used for all dual camera stereo photogrammetry image acquisition. Black and white images were captured at a rate of 250 frames per second and a resolution of 1280 x 1024 pixels. The cameras were oriented in front of the atrial chamber (Figure 4.31) and placed approximately 30˚ apart.

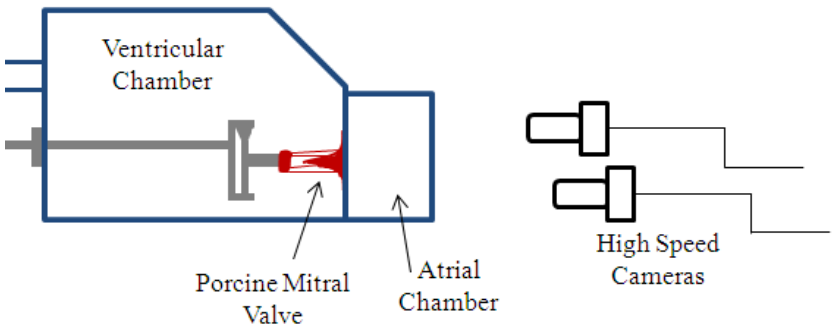


Figure 4.31 High speed camera location.

4.3.4 Echocardiography System

A Philips iE33 echocardiography system with an X7-2 ultrasound probe was used to obtain 3D full volume images of the mitral valve in static and dynamic configurations. The probe was placed on the face of the atrial chamber with ultrasound gel (Aquasonic 100, Parker Laboratories, Inc, Fairfield, NJ) in between the probe and acrylic chamber to ensure transfer of the echo signal into the chamber as depicted in Figure 4.32. The probe was kept in the same location for all conditions.

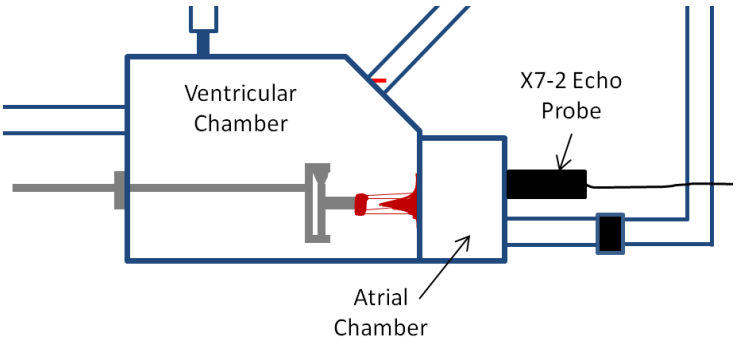


Figure 4.32 X7-2 echo probe location for dynamic experiments.

3D echocardiography images were acquired using a Philips iE33 echocardiography system (Philips Medical Systems). 3D triggered full volume images were acquired at depths of 10-14cm in 1cm increments at high and medium density. The highest quality images from each series were used for segmentation. Once the highest quality images were identified, they were transformed into Cartesian DICOMs using QLAB software (Philips Medical Systems).

4.3.5 Experimental Protocol

The protocol for the validation experiments was as follows:

1) Place mounted and tissue dye marked mitral valve in the Left Heart Simulator and position papillary muscles into a normal position

2) Adjust pressure and flow in the simulator until 5 L/min average cardiac output and 120mmHg transmitral pressure are achieved for 70 beats/min

3) Acquire 500 images at 250 frames per second triggered with the start of each cardiac cycle from both cameras simultaneously

4) Stop the left heart simulator and position the 10mm calibration cube such that 7 corners can be seen in each camera and lock in place using lock-jaw pliers

5) Acquire a single simultaneous image of the calibration cube from both frame grabbers

6) Position the X7-2 echo probe on the face of the atrial chamber so that the entire mitral orifice can be imaged

7) Acquire 3D full volume images of the calibration cube at depths of 10-14cm in 1cm increments at medium and high density settings

8) Remove the calibration cube from the atrial chamber

9) Repeat step 2

10) Acquire triggered, 3D full volume images of the mitral valve at depths of 10-14cm in 1cm increments at medium and high density. Adjust 2D opt, gain, and power to achieve the best image.

11) Stop the left heart simulator and create the diseased state. Repeat steps 1-10

4.3.6 Leaflet Marker Reconstruction

Using a direct linear transformation[40], the 3D coordinates of the leaflet markers were calculated from the 2D image pairs obtained from the high speed cameras. The high speed was triggered with the bladder pump system to synchronize acquisition with the hemodynamic data. After image collection, the *x* and *y* image coordinates of each visible marker was tracked using a marker tracking program written in MATLAB. Another program written in MATLAB was used to find the image coordinates of the 7 visible corners of a calibration cube. Then, a direct linear transformation method implemented in MATLAB was used to reconstruct the 3D spatial coordinates of each marker from the two 2D image coordinates. These vertices were then imported in to Geomagic Studio 11 (Geomagic, Inc, Research Triangle Park, NC) and triangulated using the *wrap* function.

4.3.7 Echo and Marker Time Registration

The high-speed camera and 3D echocardiography images could not be acquired simultaneously because the echo probe blocked the view of the mitral valve in both cameras. In addition, the high-speed and echocardiography images could not be triggered at the same point in the cardiac cycle due to how the echocardiography machine was triggered. The high-speed cameras were triggered from the trigger box of the left heart simulator system. The echocardiography machine was also triggered from the same trigger box, but the signal had to be transformed on the echocardiography machine in order to trigger properly. The echocardiography machine also did not trigger exactly with the falling section of the trigger signal. If the exact time from the falling section of the trigger signal could be read from the echo data set an exact time match could be found. However, the echo data set does not record this exact information and the exact time offset could not be determined.

This resulted in a time offset between the high-speed camera and echocardiography data sets that were compared. Due to this offset, the data sets needed to be registered in time. To accomplish this, the peak systolic frame from the echocardiography data and high-speed camera images were registered. The time offset between the high-speed camera images and the echocardiography images was calculated. This offset was used to match the corresponding high-speed camera and echocardiography images for valve opening and closing.

High speed camera images are taken every 4 ms and 3D echo images were taken every 21 ms. Therefore, there could be an offset up to 5 camera images (20 ms) when matching the peak systolic frames. To estimate this error, the displacement of six points in the A2 segment of the anterior leaflet were measured over 6 frames (24ms) around the opening and closing phase of each examined case. The displacement of each point was averaged and an average displacement for 5 camera frames was calculated in order to estimate the uncertainty from a time offset.

4.3.8 Marker and Virtual Model Comparison Protocols

The protocol for the cube transformation comparison was as follows:

1) Find the calibration cube origin and principal directions in the 3D echo coordinate system and apply the echo correction scheme

2) Import the PLY file of the virtual leaflet models and apply the calibration cube transformation. Export the transformed mesh to a PLY file

3) Import the 3D markers into Geomagic Studio 11 and create a triangulation between the vertices. Export as a PLY file

4) Import the 3D marker vertices into MATLAB from the PLY file

5) Import the transformed virtual model into MATLAB from the PLY file

6) Find the closest distance to the virtual model for each vertex of the marker data

7) Calculate the mean distance, standard deviation, and RMS distance the set of distances obtained for the 3D marker data

The protocol for the best fit alignment comparison was as follows:

1) Import the 3D marker coordinates into Geomagic Studio 11

2) Import the virtual model into Geomagic Studio 11

3) Perform the best-fit alignment between marker vertices and virtual model in Geomagic Studio 11

4) Generate a triangulation of the 3D marker coordinates and export as a PLY file

5) Import the 3D marker vertices into MATLAB from the PLY file

6) Import the transformed virtual model into MATLAB from the PLY file

7) Find the closest distance to the virtual model for each vertex of the marker data

8) Calculate the mean distance, standard deviation, and RMS distance the set of distances obtained for the 3D marker data

4.3.9 *In vitro* Correction for 3D Echocardiography

It is assumed by the echo machine that the ultrasound beam travels through a homogenous medium with a speed of sound of 1540 m/s. This assumption is normally sufficient for *in vivo* measurements, but the *in vitro* flow loop used in this study contains multiple media (ultrasound gel, acrylic plate, saline) between the probe and the mitral valve. Each of these media has a varying speed of sound. The resulting differences in speed of sound and refraction at media interfaces resulted in errors in the 3D echocardiography measurements. In an effort to account for these errors, a correction scheme was devised.

Snell's law defines how sound waves refract when passing between two mediums with different speeds of sound. An example of this is shown in Figure 4.33 and the equation for this relationship is defined in Equation 4.12[41, 42].

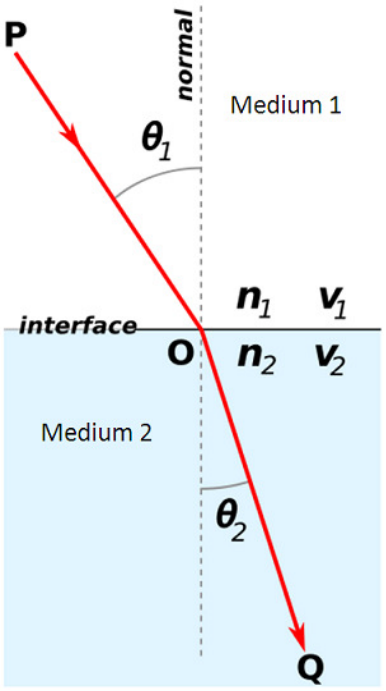


Figure 4.33 Example of Snell's Law (modified from http://www.math.cornell.edu/~numb3rs/lundell/snellimage.png).

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{v_1}{v_2} \tag{4.12}$$

where:

v_1*: speed of sound in medium 1*

v_2*: speed of sound in medium 2*

θ_1*: incident angle from normal in medium 1*

θ_2*: incident angle from normal in medium 2*

A correction was determined in 3D spherical coordinates for the *in vitro* model using Snell's Law and the speed of sound for each of the three media. Given the measured spherical coordinates and measured time of travel of a point in the virtual model, the true coordinate of the point could be determined using Snell's Law and calculating the time of travel in each medium. The three media that the sound beam passes through, in order, are ultrasound gel, the acrylic plate of the atrial chamber, and saline. The speed of sound of each medium is shown in Table 4.1.

Table 4.1 Speed of sound in various media

Medium	Speed of Sound
Ultrasound Gel	1540 m/s[43]
Acrylic	2870 m/s[44]
Saline	1502 m/s[45]

An exaggerated 2D example of how the sound beam could travel in the *in vitro* setup is shown in Figure 4.34.

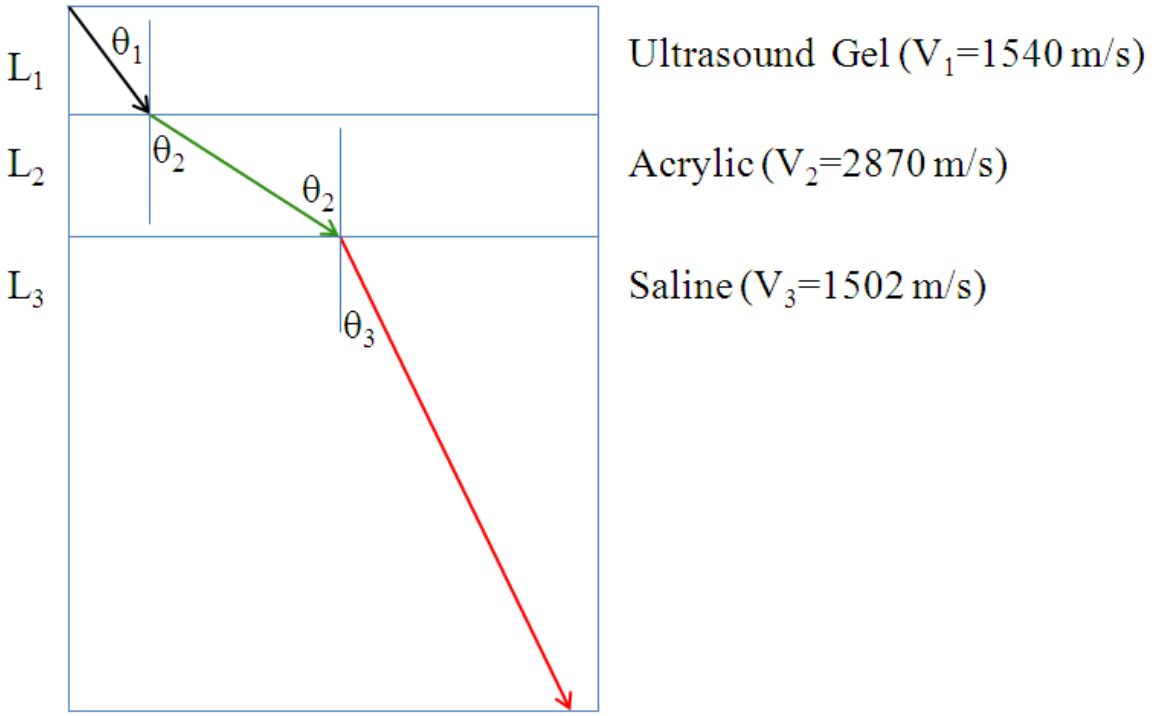
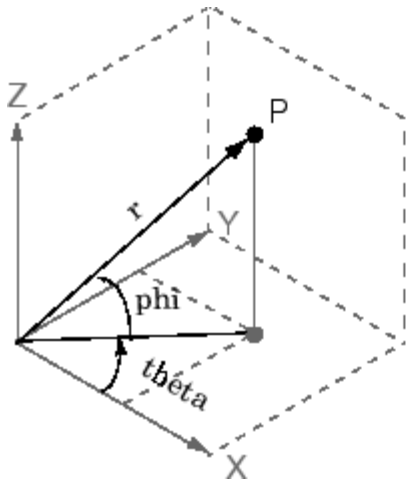


Figure 4.34 2D Refraction of sound through 3 media (ultrasound gel, acrylic, saline).

To extend this to 3D, a spherical coordinate system was used and is defined in Figure 4.35, where the *x*-direction was defined as the depth and *y* and *z* are the lateral directions.




```
x = r .* cos(phi) .* cos(theta)
y = r .* cos(phi) .* sin(theta)
      z = r .* sin(phi)
```

Figure 4.35 Spherical coordinate system used for echo correction with Cartesian coordinate system for reference (http://www.mathworks.com/access/helpdesk/help/techdoc/ref/math_s19.gif).

Using this coordinate system, the problem of calculating the refraction and length traveled within a particular medium is shown in Figures 4.36 and 4.37.

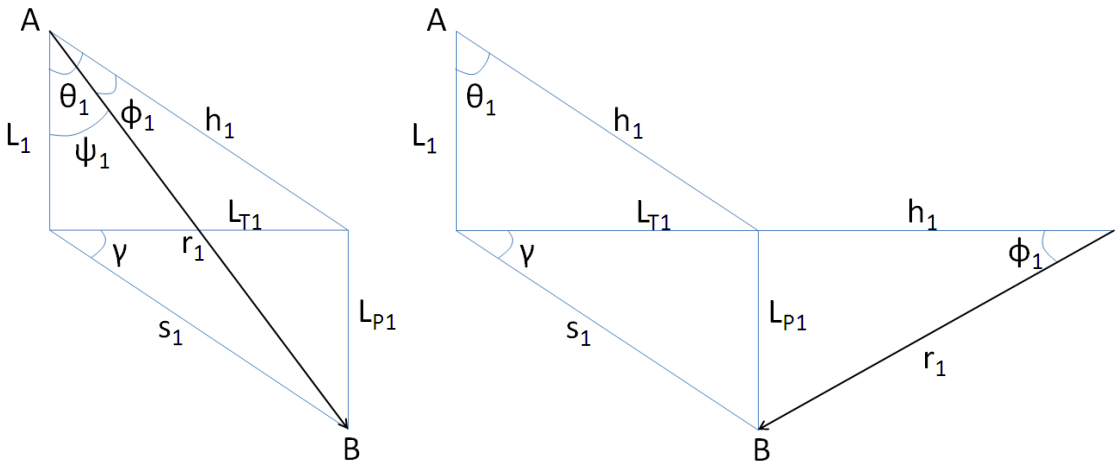


Figure 4.36 Beam traveling from *A* to *B* in first medium using spherical coordinates (bold = beam path).

where:

A: starting position of the beam

B: point at which the beam meets the second medium

r_1: radius of the beam in medium 1

θ_1: azimuthal angle of the beam in medium 1

φ_1: polar angle of the beam in medium 1

L_1: thickness of medium 1

ψ_1: angle of incidence from normal for refraction for medium 1

L_{T1}, L_{P1}, s_1, h_1: lengths of their respective sides

γ: angle between L_{T1} and s_1

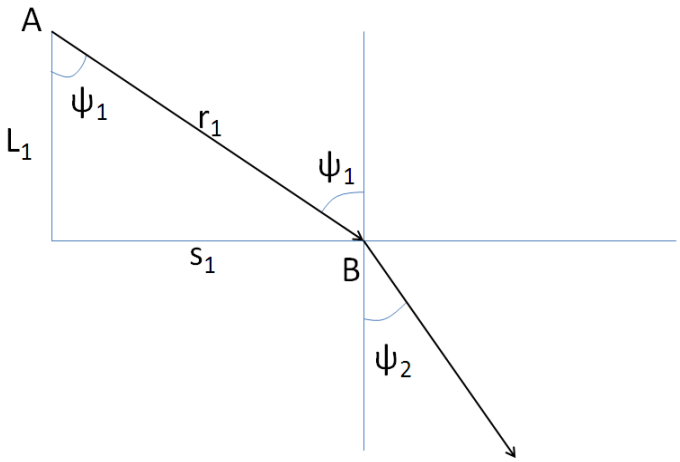


Figure 4.37 Refraction plane for beam traveling from A to B from first to second medium using spherical coordinates (bold = beam path).

where:

A: starting position of the beam

B: point at which the beam meets the second medium

r_1: radius of the beam in medium 1

L_1: thickness of medium 1

ψ_1: angle of incidence from normal for refraction for medium 1

ψ_2: angle of incidence from normal for refraction for medium 2

s_1: length of its respective side

The formulae necessary to determine the refraction angles, ψ_1 and ψ_2, are as follows:

$$L_T = L \tan\theta \tag{4.13}$$

$$\cos\gamma = \frac{L_T}{s} = \frac{L \tan\theta}{s} \tag{4.14}$$

$$h = \frac{L}{\cos\theta} \tag{4.15}$$

$$L_p = h \tan\varphi = L \frac{\tan\varphi}{\cos\theta} \tag{4.16}$$

$$s = \sqrt{L_T^2 + L_p^2} = L \sqrt{\tan^2\theta + \frac{\tan^2\varphi}{\cos^2\theta}} \tag{4.17}$$

$$\gamma = \cos^{-1}\left(\frac{\tan\theta}{\sqrt{\tan^2\theta + \frac{\tan^2\varphi}{\cos^2\theta}}} \right) \tag{4.18}$$

$$\psi_1 = \tan^{-1}\frac{s}{L} = \tan^{-1}\sqrt{\tan^2\theta + \frac{\tan^2\varphi}{\cos^2\theta}} \tag{4.19}$$

$$\psi_2 = \sin^{-1}\left(\frac{v_2 \sin\psi_1}{v_1}\right) \tag{4.20}$$

where Equations 4.19-4.20 are necessary to determine the refraction into the next medium. As the beam refracts into the second medium, spherical coordinates of the path need to be determined. This problem is shown in Figures 4.38 and 4.39 where θ_2 and φ_2 are unknown, L_2 is the thickness of the second medium, and ψ_2 is the refraction angles for the second medium. All other variables listed in the figures were used in calculating θ_2 and φ_2 in the second medium.

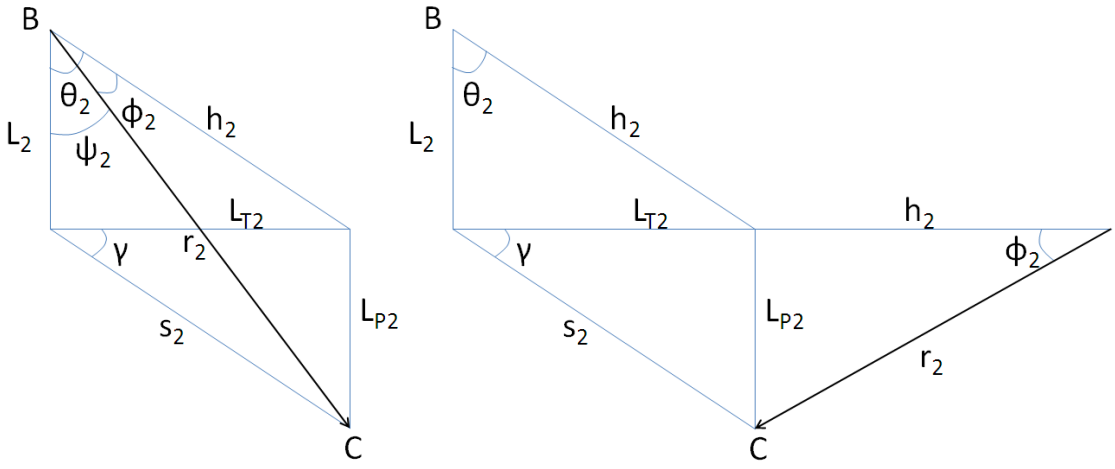


Figure 4.38 Beam traveling from B to C in second medium using spherical coordinates (bold = beam path).

where:

B: point at which the beam meets the second medium

C: point at which the beam meets the third medium

r_2: radius of the beam in medium 2

θ_2: azimuthal angle of the beam in medium 2

φ_2: polar angle of the beam in medium 2

L_2: thickness of medium 2

ψ_2: angle of incidence from normal for refraction for medium 1

L_{T2}, L_{P2}, s_2, h_2: lengths of their respective sides

γ: angle between L_{T2} and s_2

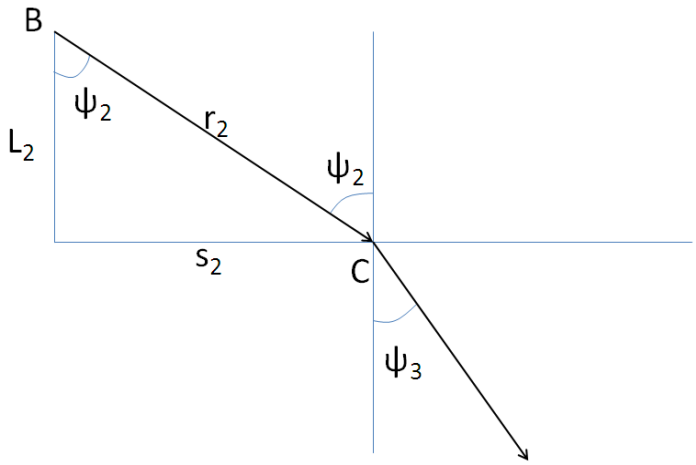


Figure 4.39 Refraction plane for beam traveling from B to C from second to third medium using spherical coordinates (bold = beam path).

where:

B: point at which the beam meets the second medium

C: point at which the beam meets the third medium

r_2: radius of the beam in medium 2

L_2: thickness of medium 2

ψ_2: angle of incidence from normal for refraction for medium 2

ψ_3: angle of incidence from normal for refraction for medium 3

s_2: length of its respective side

θ_2 and φ_2 are calculated by,

$$s_2 = L_2 \tan \psi_2 \tag{4.21}$$

$$r_2 = \frac{L_2}{\cos \psi_2} \tag{4.22}$$

$$L_{T2} = s_2 \cos \gamma = L_2 \tan \psi_2 \cos \gamma \tag{4.23}$$

$$L_{P2} = s_2 \sin \gamma = L_2 \tan \psi_2 \sin \gamma \tag{4.24}$$

$$\theta_2 = \tan^{-1}\left(\frac{L_{T2}}{L_2}\right) = \tan^{-1}(\tan \psi_2 \cos \gamma) \tag{4.25}$$

$$\sin \varphi_2 = \frac{L_{P2}}{r_2} = \tan \psi_2 \cos \psi_2 \sin \gamma = \sin \psi_2 \sin \gamma \tag{4.26}$$

$$\sin \psi_2 = \frac{v_2 \sin \psi_1}{v_1} \tag{4.27}$$

$$\sin \varphi_1 = \frac{L_{P1}}{r_1} \tag{4.28}$$

$$\sin\varphi_2 = \sin\psi_2 \sin\gamma = \frac{v_2 \sin\psi_1}{v_1}\sin\gamma = \frac{v_2}{v_1}\frac{s_1}{r_1}\frac{L_{P1}}{s_1} = \frac{v_2}{v_1}\frac{L_{P1}}{r_1} = \frac{v_2 \sin\varphi_1}{v_1} \tag{4.29}$$

$$\varphi_2 = \sin^{-1}\frac{v_2 \sin\varphi_1}{v_1} \tag{4.30}$$

Using the Equations 4.18-4.20, 4.25 and 4.30, the beam path can be determined as it travels between mediums. This can be iteratively solved to determine the beam path through 3 different mediums. It should be noted that Equation 4.30 is Snell's Law for φ_1 and φ_2.

For the echo correction, the initial spherical coordinates were given by the echo measurement. From this, the total time of travel could be determined. The thickness of the ultrasound gel was assumed to be very thin (0.01cm) and the thickness of the acrylic plate was measured. The script for echo correction initially calculated the time of travel in the ultrasound gel based upon the angles provided by the echo measurement and the coordinates of the interface between the ultrasound gel and the acrylic plate. Then the refraction into the acrylic was calculated. The new spherical coordinates of the beam direction were calculated along with the time of travel in the acrylic plate. The location of the interface between the acrylic plate and the saline was also calculated. Finally, the time of travel in the ultrasound gel and the acrylic plate were subtracted from the given total time of travel. The final actual coordinate of the point measured was determined and the point coordinates were updated in the virtual model. The MATLAB code for the echo correction of a single point is as follows:

```
% Define variables of in vitro setup
tU = 0.01;          % thickness of US gel in cm
tA = 0.821;         % thickness of acrylic plate in cm
tS = 5.137;         % thickness of saline volume in cm
sU = 1540*100;      % speed of sound of US gel in cm/s
sA = 2870*100;      % speed of sound of acrylic in cm/s
sS = 1502.3*100;    % speed of sound of saline in cm/s
```

```
% Define variable needed to correction
pt; % single point in the original echo coordinate system
ptT = (pt-orig)*0.5/10;  % convert coordinates to cm
ptT = [ptT(2)  ptT(3) ptT(1)];  % swap coords so x value = depth
[theta,phi,origR] = cart2sph(ptT(1),ptT(2),ptT(3)); % calc spherical
measuredTime = (origR)/sU;  % calc time measured by echo

% Calculate distance and time in ultrasound gel
rU = tU/(cos(phi)*cos(theta)); % calc radius in ultrasound gel
timeU = rU/sU; % calc time spent in ultrasound gel
psi = atan(sqrt((tan(phi)^2)/(cos(theta)^2)+tan(theta)^2)); % calc psi
gamma = acos(tan(theta)/tan(psi)); % calc gamma
[xU,yU,zU] = sph2cart(theta,phi,rU); % calc medium transition point
pU = [xU yU zU]; % store Cartesian coord

% Calculate distance and time in acrylic
psiA = asin(sA*sin(psi)/sU); % calc psi in acrylic
rA = tA/cos(psiA); % calc radius in acrylic
timeA = rA/sA; % calc time spent in acrylic
thetaA = atan(tan(psiA)*cos(gamma)); % calc theta in acrylic
phiA = asin(sA*sin(phi)/sU); % calc phi in acrylic
[xA,yA,zA] = sph2cart(thetaA,phiA,rA); % calc medium transition point
pA = [xA yA zA]; % store Cartesian coord (relative to gel layer)

% Calculate distance and time in saline (final medium)
psiS = asin(sS*sin(psiA)/sA); % calc psi in saline
timeS = measuredTime-timeU-timeA; % time beam in saline
rS = sS*timeS; % determine radius from time and speed of sound
thetaS = atan(tan(psiS)*cos(gamma)); % calc theta in saline
phiS = asin(sS*sin(phiA)/sA); % calc phi in saline
[xS,yS,zS] = sph2cart(thetaS,phiS,rS); % calc Cartesian point measured
pS = [xS yS zS]; % store Cartesian coord (relative to acrylic layer)

% Calculate final Cartesian coordinate
pFinal = pU + pA + pS;
```

where:

tU,tA,tS: thickness of ultrasound gel (tU), acrylic (tA), and saline (tS) layers [cm]

sU,sA,sS: speed of sound in ultrasound gel (sU), acrylic (sA), and saline (sS) [cm/s]

pt: original point in echo coordinates

orig: origin of the ultrasound beam relative to the echo coordinates

ptT: original point with orig set as the origin and scaled to cm

theta, phi, origR: spherical coordinates from echo coordinates

measuredTime: time measured by echo

rU,rA,rS: radius of beam in ultrasound gel (rU), acrylic (rA), and saline (rS) [cm]

timeU,timeA,timeS: time of beam in ultrasound gel (timeU), acrylic (timeA), and saline (timeS) [s]

psi,psiA,psiS: refraction angles for ultrasound gel (psi), acrylic (psiA), and saline (psiS)

thetaA,thetaS: theta of spherical coordinates for acrylic (thetaA) and saline (thetaS)

phiA,phiS: phi of spherical coordinates for acrylic (phiA) and saline (phiS)

pU: Cartesian point at boundary layer between ultrasound gel and acrylic

pA: Cartesian point at the boundary layer between acrylic and saline relative to the ultrasound gel layer

pS: Cartesian point of the actual beam location in saline relative to the acrylic layer

pFinal: corrected location relative to the ultrasound probe

This MATLAB code was applied to each vertex in the virtual model in order to correct the entire virtual model for the given atrial chamber used.

To validate the echo correction scheme, an annulus plate with parallel edges near the annular orifice was constructed (Figure 4.40). This annulus plate was imaged using multiple atrial chambers with varying wall thicknesses. The specifications for each chamber are shown in Table 4.2. The errors before and after the correction were compared.

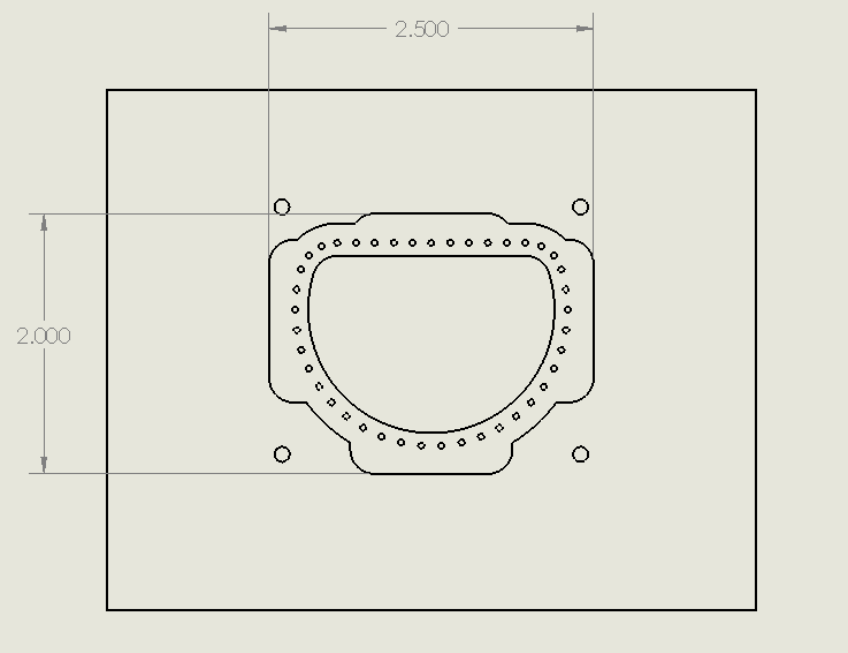


Figure 4.40 Annulus plate for echo correction scheme validation.

Table 4.2 Atrial chamber measurements

	Chamber 1	Chamber 2	Chamber 3
Ultrasound Gel Layer (mm)	0.1	0.1	0.1
Acrylic Layer (mm)	8.21	12.32	2.97
Saline Layer (mm)	51.37	38.10	50.80

4.3.10 Transformation and Best Fit between Echo and Marker Coordinate Systems

To determine the transformation between the echo and marker coordinate systems, the vectors for each edge of the calibration cube (principal vectors) were determined in the echo coordinate system using a MATLAB program. The vectors were then tested for orthogonality after echo correction was applied. These vectors were found to be slightly non-orthogonal due to the echocardiography resolution of 0.5mm. To complete the transformation, the echo basis was defined with e.o as the origin. Also. e.x, e.y, and e.z were defined as the principal directions of the echo basis. The cube basis was defined with an origin of c.o, and c.x, c.y, and c.z were defined as the principal directions of the calibration cube in the echo coordinate system. To transform a coordinate, p, in the echo basis to a coordinate in the cube basis, p', the following formulae were used:

$$v = p - c.o \tag{4.31}$$

where:

p: point p in the echo coordinate system

c.o: origin of the calibration cube in the echo coordinate system

v: vector between points c.o and p

$$x = \frac{v \cdot (c.y \times c.z)}{c.x \cdot (c.y \times c.z)} \tag{4.32}$$

$$y = \frac{v \cdot (c.x \times c.z)}{c.y \cdot (c.x \times c.z)} \tag{4.33}$$

$$z = \frac{v \cdot (c.x \times c.y)}{c.z \cdot (c.x \times c.y)} \tag{4.34}$$

where:

x: scaling factor for the x-direction

y: scaling factor for the y-direction

z: scaling factor for the z-direction

v: vector between points c.o and p

c.x: principal x-direction defined on the calibration cube

c.y: principal y-direction defined on the calibration cube

c.z: principal z-direction defined on the calibration cube

To calculate the point *p'* that has been transformed into the cube basis, Equation 4.35 is used.

$$p' = e.o + x * e.x + y * e.y + z * e.z \tag{4.35}$$

where:

p': point p transformed to the cube basis

e.o: echo coordinate system origin

e.x: echo coordinate system principal x-direction vector

e.y: echo coordinate system principal y-direction vector

e.z: echo coordinate system principal z-direction vector

x: scaling factor for the x-direction

y: scaling factor for the y-direction

z: scaling factor for the z-direction

A script was written in MATLAB for the cube transformations performed using this method. This script transformed each vertex of the virtual model from the echo coordinate system to the calibration cube coordinate system using the coordinates of the calibration cube origin and edges measured from the 3D echo.

In addition to the cube transformation, a best fit method was used to compare the marker data to the virtual model. The best fit algorithm in Geomagic Studio 11 was used to obtain a best fit between the marker data and the virtual model surface. This method minimized square distances between the marker points and the virtual model surface.

4.3.11 Distance Measurement between Marker Data and Virtual Model

To quantify the match between the virtual model and reconstructed markers, the distance between each marker vertex from the virtual model was measured. Some offset is expected because the markers are placed on the atrial surface of the mitral leaflets and the segmentation was performed on the middle of the leaflets. To find the closest distance between a marker point and the virtual model, the following steps are necessary for each marker point:

1) Find the closest vertex in the virtual model to the marker point.

2) Search all edges of the virtual model and determine if there is a point along any edge that is closer than the closest vertex.

3) Search all triangles of the virtual model and determine if there is a point in any triangle that is closer than the closest vertex and edge.

Equation 4.36 was used to determine the closest vertex in the virtual model to each marker point.

$$d = \left((v_x - m_x)^2 + \left(v_y - m_y\right)^2 + (v_z - m_z)^2\right) = |mv| \tag{4.36}$$

where:

v_i*: i-coordinate of the virtual model vertex*

m_i*: i-coordinate of the marker vertex*

d: distance between the vertices v and m

Then for each marker vertex, the closest edge of the virtual model was found (Figure 4.41). The minimum distance was updated if the distance between the marker vertex and the closest edge was less than that between the marker data and the closest virtual model vertex. Given the line segment *ab* defined between points *a*, *b*, and point *p* in 3D Cartesian space, the distance between point *p* and *ab* is defined in the following formulas.

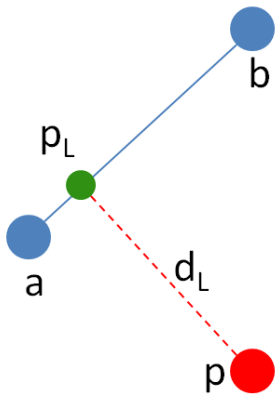


Figure 4.41 Distance (d_L) between a line segment (ab) and a point (p) with point of intersection (p_L).

Let ab be the vector between a and b, ba be the vector between b and a, ap be the vector between a and p, and bp be the vector between b and p. Let d_L be the distance between the point p and the line segment ab. There are three conditions that need to be evaluated to determine d_L [46, 47]. There are:

1) If $ab \cdot bp >= 0$, then $d_L = |bp|$, endpoint b is closest to p

2) If $ba \cdot ap >= 0$, then $d_L = |ap|$, endpoint a is closest to p

3) If $ab \cdot bp < 0$ and $ba \cdot ap < 0$, d_L is defined in Equation 4.37

$$d_L = \frac{|ap \times bp|}{|ab|} \tag{4.37}$$

Equation 4.38 can be used to find the closest point, p_L, on the line segment ab to p[46, 47].

$$p_L = p - d_L \left[\frac{(ab \times ap) \times ab}{|(ab \times ap) \times ab|} \right] \tag{4.38}$$

The overall minimum distance between the marker vertices and the virtual model vertices and edges were computed. Next, the virtual model triangles were searched to determine if a smaller distance between the marker vertices and the triangle could be found. Given a triangle *abc* defined by the points *a*, *b*, and *c* and a marker vertex, *p*, the distance to the plane created by the triangle was found using the following formulae (4.39-4.41).

$$t = \frac{a + b + c}{3} \tag{4.39}$$

where:

t: barycenter (centroid) of the triangle abc

a,b,c: vertices of triangle abc

$$u = \frac{ab \times ac}{|ab \times ac|} \tag{4.40}$$

where:

u: unit normal of the plane defined by triangle abc

Given *t* and *u,* the distance between the point *p* and the plane of triangle *abc* can be found using Equation 4.41.

$$d_T = |u \cdot tp| \tag{4.41}$$

where:

tp: vector between p and t (barycenter of the triangle)

u: unit normal of the plane defined by triangle abc

The coordinate of the closest point to the plane, p_T, is dependent on the side of the triangle that the marker vertex, *p*, is on. If $tp \cdot u <= 0$, then the triangle normal, *u*, and the vector between *p* and *t* are in opposite directions. Therefore, p_T is defined by Equation 4.42.

$$p_T = p + d_T * u \tag{4.42}$$

If $tp \cdot u > 0$, then the triangle normal, *u*, and the vector between *p* and *t* are in the same direction. Therefore, p_T is defined by Equation 4.43.

$$p_T = p - d_T * u \tag{4.43}$$

where:

p_T: closest point on the plane to p

d_T: distance to closest point

u: triangle unit normal

Next, it must be determined if the closest point on the plane, p_T, is inside the triangle *abc*. This can be accomplished by using Equations 4.44 and 4.45 to calculate *u* and *v* respectively. If *u*>0, *v*>0, and *u*+*v*<1, then p_T is in the triangle *abc*, otherwise it is outside the triangle *abc*[48].

$$u = \frac{ab \cdot ab * ac \cdot ap_T - ac \cdot ab * ab \cdot ap_T}{ac \cdot ac * ab \cdot ab - ac \cdot ab * ac \cdot ab} \tag{4.44}$$

$$v = \frac{ac \cdot ac * ab \cdot ap_T - ac \cdot ab * ac \cdot ap_T}{ac \cdot ac * ab \cdot ab - ac \cdot ab * ac \cdot ab} \tag{4.45}$$

where:

ab: vector between vertex a and b of triangle abc

ac: vector between vertex a and c of triangle abc

ap_T: vector between vertex a of triangle abc and point p_T

If p_T is inside the test triangle and is closer than the previously calculated distance from the marker vertex to the edges of the virtual model, then that position is updated as a final position.

$$v = \frac{ac \cdot ac * ab \cdot ap_T - ac \cdot ab * ac \cdot ap_T}{ac \cdot ac * ab \cdot ab - ac \cdot ab * ac \cdot ab} \tag{4.45}$$

where:

ab: vector between vertex a and b of triangle abc

ac: vector between vertex a and c of triangle abc

ap_T: vector between vertex a of triangle abc and point p_T

4.3.12 Leaflet and Coaptation Area Calculations

The virtual models created in this study are triangular meshes. These meshes can be used to calculate additional measures from the echocardiography data sets. These include leaflet area and coaptation area. The area of each leaflet can then be found by

summing the areas of all of the triangles that make up the entire leaflet mesh. The area of each triangle in the mesh was determined by using Equation 4.46[49], which calculates the area of a triangle in 3D space given the coordinates of each of its vertices.

$$Triangle\ Area = \frac{1}{2}|(b-a) \times (a-c)| \tag{4.46}$$

where:

a,b,c: vertices of the triangle abc

To determine the coaptation area from the two leaflet meshes, the vertices in each mesh that were within 1.26 mm of each other were found[50]. Then the area covered by these vertices was calculated using Equation 4.46.

4.3.13 Statistical Analysis

Three formulae were used to analyze the calculated distances between the marker data and the virtual models. These were the mean distance, standard deviation of distance, and the root mean square (RMS) distance that are defined in Equations 4.46, 4.47, and 4.48, respectively.

$$mean = \sum_{i=1}^{n} \frac{d_i}{n} \tag{4.46}$$

$$standard\ deviation = \frac{1}{n}\sum_{i=1}^{n}(d_i - m)^2 \tag{4.47}$$

$$RMS\ Distance = \frac{1}{n}\sum_{i=1}^{n} {d_i}^2 \tag{4.48}$$

where:

n: number of points

d_i: ith distance

m: mean of the distances

The mean distance is the average absolute distance between the reconstructed marker vertices and the corresponding virtual model. The standard deviation measures the deviation in the distances between marker data and the virtual model. The RMS distance measures the overall fit between the markers and the virtual model. For each of these measures, lower values indicate better shape match.

4.3.14 Error Analysis

There were four sources of error in the validation method used in this study. Two were systematic errors and two were random errors. The two sources of systematic error were error in the 3D reconstruction of the marker data and the error from the offset of the segmentation and the marker data. Errors in the 3D reconstruction are a limitation of the DLT method. The larger the distance between the calibration cube and the markers, the larger the error will be. This error can be estimated by reconstructing points along the suturing holes of the annulus plate. In reality these points must form a perfect plane, but after 3D reconstruction these points do not fall upon a single plane. To estimate this error a plane was created between the reconstructed annulus points closest to the calibration

cube. After this the distance between each of the points and the plane was found using Equation 4.41 from section 4.3.11. This distance was used as the estimate for how the error from the 3D reconstruction changed across the valve.

Segmentations were performed on the middle of the valve because the thickness of the valve can be affected by the gain settings on the machine. However, the markers were placed on the atrial side of the leaflets. This resulted in an offset between the segmented data and the marker data. This offset was a result of the thickness of the mitral valve. While the thickness of each leaflet varies across each leaflet it was assumed that both leaflets had an average constant thickness of approximately 1.26 mm[50]. This thickness resulted in an expected offset between the marker data and the virtual models of 0.63 mm. This represents the systematic error that occurs as a result of the validation method.

The two random errors associated with the validation method were the resolution of the echocardiography data set and the time mismatch that could arise from the method of matching the echocardiography and high speed images in time. The echocardiography data had a resolution of 0.5 mm. If the exact middle of the leaflet is assumed to be within the selected pixel, then the uncertainty for each selected point is ± 0.25 mm. Since the exact uncertainty of the time mismatch cannot be determined, it will be estimated for each case of valve opening and closing. It will be estimated by finding the average displacement of 6 markers in the belly region of the anterior leaflet over 6 surrounding high speed frames. It will also be estimated near the commissures by finding the average displacement of 6 markers in the commissural region of the leaflets over 6 surrounding high speed frames. The error will then be estimated by determining the average leaflet displacement over 3 frames to determine the amount of uncertainty associated with the time mismatch.

Given the two sources of systematic error the total systematic error will be estimated by adding the error from the segmentation offset (*A*) and the errors calculated for the 3D reconstruction (*B*).

$$Error_{Systematic} = A + B \tag{4.49}$$

where:

A: error from segmentation offset

B: error from 3D reconstruction

The errors from the 3D reconstruction were higher near the commissures because those markers are farthest away from the calibration cube. Therefore, the systematic error was calculated separately for the commissural region and the belly region of the leaflet.

The total random error was estimated by finding the total uncertainty from the echocardiography resolution and the time mismatch. This is shown in Equation 4.50.

$$Error_{Random} = \sqrt{C^2 + D^2} \tag{4.50}$$

where:

C: uncertainty resulting from the resolution of the echocardiography data set

D: uncertainty resulting from the time mismatch between echocardiography and high-speed data sets

Since the valve speed in the belly region of the leaflet varies significantly from the valve speed in the commissural region of the leaflet, the random error was calculated separately for the commissural region and the belly region of the leaflet. Therefore, for each case examined, a total random and systematic error was calculated for both the commissural and belly region of the leaflets.

The values of each of these errors (*A,B,C,D*) will be calculated for each validation data set. For peak systolic cube transformation and best fit cases, there will be negligible uncertainty from the time mismatch (D=0) because the leaflet motion around the surrounding frames is negligible. For all of the best fit cases, there will be no offset associated with the calculation (A=0). This is based on the assumption that if the shapes of the markers and the virtual model are the same, then a best fit alignment between the data sets would result in no offset between them.

CHAPTER 5

RESULTS

5.1 Overview

The results of this study are divided into four sections. Section 5.2 contains results from specific aim 1, which was to develop a robust tool to generate virtual leaflet models from real-time 3D echocardiography data. Sections 5.3 and 5.4 focus on the results from specific aim 2, which was to validate the virtual mitral valve leaflet models with a dynamic in vitro mitral valve model. The final section, 5.5, corresponds to the results from specific aim 3, which was to investigate interpolating triangular mesh refinement methods to decrease segmentation time and determine the method that best interpolates the mitral leaflets using a dynamic in vitro model. Section 5.2 demonstrates and compares each segmentation technique. Section 5.3 focuses on the results from the 3D *in vitro* echo correction scheme that was developed in the second specific aim. Section 5.4 concentrates on the results from the dynamic valve experiments. Section 5.5 focuses on the results from the mesh refinement techniques that are aimed at reducing the number of segmented images needed to reconstruct the valve geometry.

5.2 Segmentation Methods

Point, line, and curve segmentation methods were evaluated. The segmentation methods were compared by applying each methodology to the anterior leaflet of the same segmentation slice. Figure 5.1 shows the results from each of the segmentation methods.

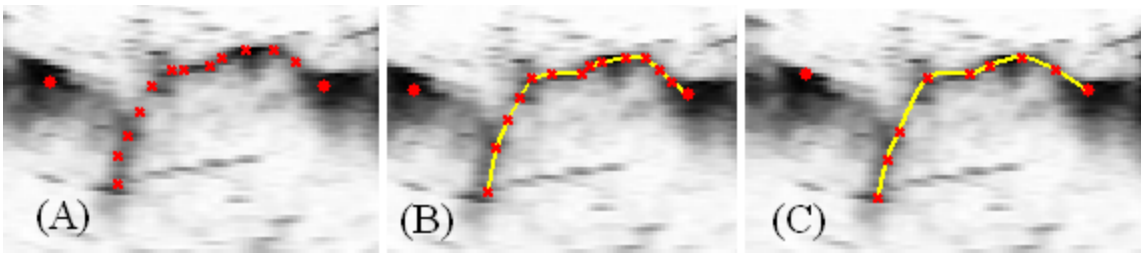


Figure 5.1 Results of segmentation of the anterior leaflet of the MV in the same slice using (A) point, (B) line, and (C) curve segmentation methods.

It can be seen in Figure 5.1A that the point segmentation method relies on the user estimating how many points are required to segment the leaflet geometry. This can result in too many or too few points selected to accurately represent the leaflet geometry. Using this method it took approximately 1 hour to segment one time point of the mitral valve. In Figure 5.1B, the line segmentation provides a guide for how many points will be required to segment the leaflet and reduces the number of points the user needs to select. Using this method it took approximately 40 minutes to segment one time point of the mitral valve. In the final method (Figure 5.1C), the use of curves further reduces the number of user-selected points needed to segment the leaflet. Using this method it took approximately 20 minutes to segment one time point of the mitral valve. The curves based method was used for all segmentations.

5.3 Three-Dimensional *in vitro* Echo Correction

Results from the 3D echo correction scheme will be presented in two parts: theoretical and experimental. Section 5.3.1 presents a theoretical analysis of the change in beam angle and beam radius errors when varying beam angle, beam radius, and acrylic thickness. Section 5.3.2 contains the results from the experimental evaluation of the echo correction scheme applied to three atrial chambers of varying thickness.

5.3.1 Echo Correction Theoretical Results

When acquiring echocardiography images from an *in vitro* environment, the speed of sound of each media and the resulting refraction must be taken into account. The resulting errors in echocardiography measurements vary depending on beam angle, beam radius, and media properties (speed of sound, thickness). The isolated effect of each of these on the beam angle and radius errors was theoretically evaluated for the experiment setup used in this study. Figures 5.2, 5.3, and 5.4 depict the effect of beam angle, beam radius, and acrylic thickness, respectively. Figure 5.2 illustrates the variance in the beam angle and radius errors for a 5.5 cm beam radius with an incident angle from 0° to 30° in atrial chamber 3. A beam radius of 5.5 cm was used because that is the approximate radius at which the valve measurements occurred. Figure 5.3 shows the effect of varying beam radius from 1 to 7 cm for an ultrasound beam traveling through atrial chamber 3 with a 15° incident angle. Figure 5.4 illustrates the effect of the acrylic layer thickness on the angle and radius errors for an ultrasound beam with a 15° incident angle and 5.5 cm radius.

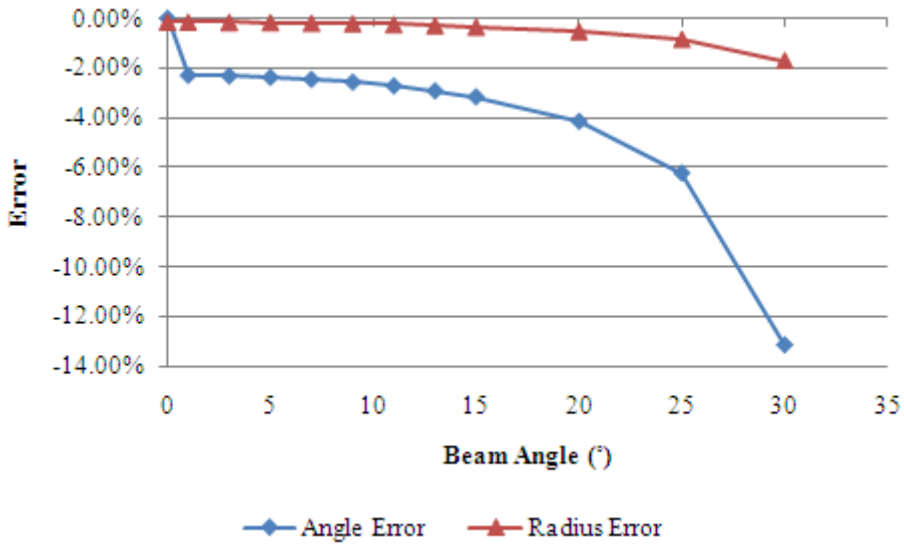


Figure 5.2 Angle and radius errors of an ultrasound beam with a 5.5 cm radius with an incident angle from 0° to 30° that passes through atrial chamber 3, which was used for all *in vitro* experiments.

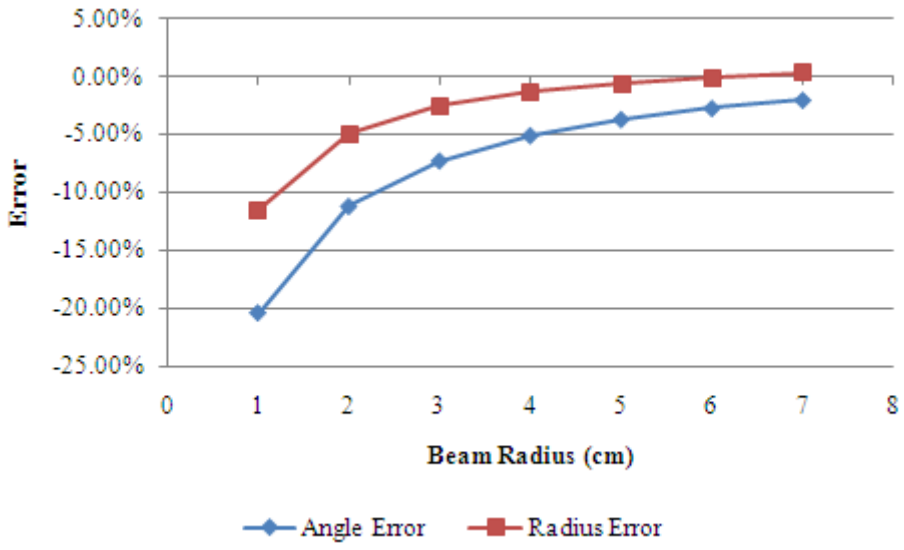


Figure 5.3 Angle and radius error for an ultrasound beam with a 15° incident angle and a beam radius from 1 to 7 cm that passes through atrial chamber 3, which was used for all *in vitro* experiments.

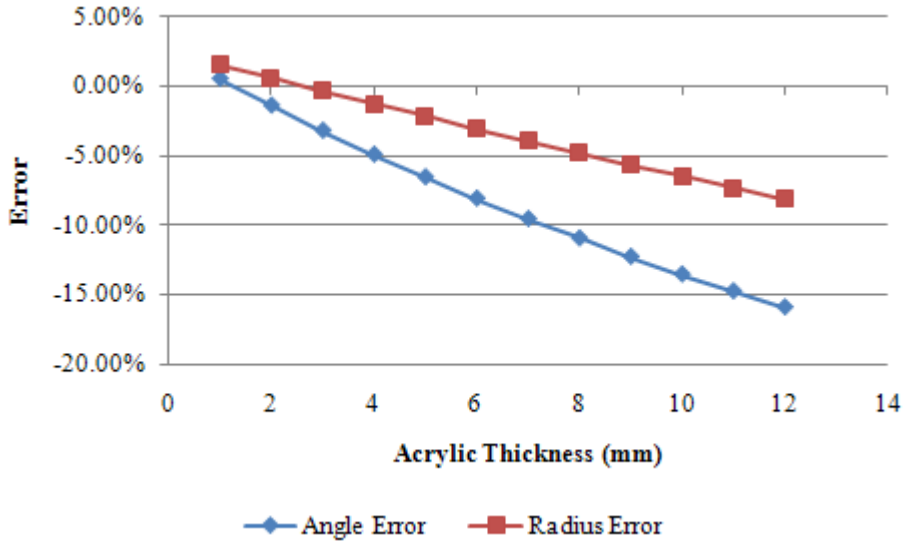


Figure 5.4 Angle and radius error for an ultrasound beam with a 15˚ incident angle and a beam radius of 5.5 cm that passes through an atrial chamber with an acrylic plate of thickness from 1 to 12 mm.

Figure 5.2 shows that as the beam angle is varied from 0° to 30°, the beam radius error is less than 2% for all cases up to 30˚. However, the angle error grows exponentially with the incident angle. Figure 5.3 shows that as the beam radius varies from 1 to 7 cm, both the angle and radius errors decrease exponentially. Figure 5.4 shows an approximately linear increase in angle and radius error as acrylic thickness increased.

Together, Figures 5.2 to 5.4 illustrate the importance of applying a correction scheme to *in vitro* ultrasound measurements. Echo measurements taken through a relatively thin (2.97 mm) piece of acrylic require a correction scheme to increase the accuracy of the measurement. In addition, the error is non-linear across the measured space and increases linearly with acrylic thickness. Therefore, the complex error induced by an *in vitro* setup means that any *in vitro* echo measurements used to quantify geometry or position should use a correction scheme to account for the speed of sound and refraction.

5.3.2 Echo Correction Experimental Results

The 3D echo correction described in section 4.3.8 was evaluated using echo images of an annulus plate (Figure 5.5) acquired using three different atrial chambers. The three chambers had varying acrylic and saline depth (Table 5.1). The error was calculated before and after the echo correction scheme was applied to the data set. The results from the echo correction are displayed in Table 5.2.

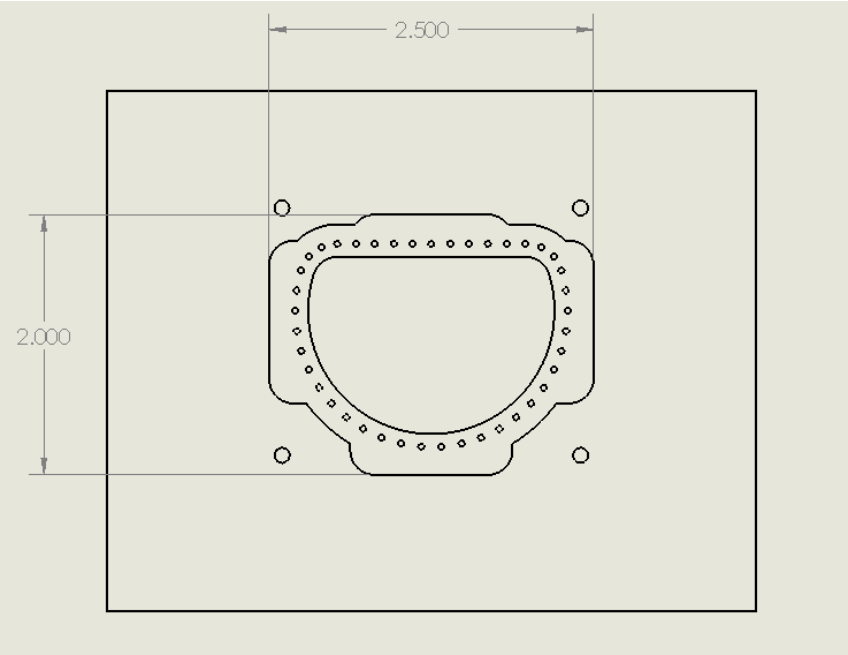


Figure 5.5 Annulus plate schematic depicting the horizontal and vertical dimensions measured in the echo correction scheme evaluation.

Table 5.1 Atrial chamber dimensions

	Chamber 1	Chamber 2	Chamber 3
Ultrasound Gel Layer (cm)	0.01	0.01	0.01
Acrylic Layer (cm)	0.82	1.23	0.30
Saline Layer (cm)	5.13	3.81	5.08

Table 5.2 Comparison of annulus plate measurements before and after correction using three different atrial chambers

		Actual (cm)	Measured (cm)	After Correction (cm)	Measured Error	Error After Correction
Atrial Chamber 1	Horizontal	6.320	5.15	6.34	-18.51%	0.32%
	Vertical	5.044	4.15	5.16	-17.73%	2.29%
Atrial Chamber 2	Horizontal	6.320	4.45	6.38	-29.58%	0.96%
	Vertical	5.044	3.65	4.95	-27.64%	-1.87%
Atrial Chamber 3	Horizontal	6.320	5.80	6.31	-8.22%	-0.15%
	Vertical	5.044	4.75	4.93	-5.84%	-2.27%

Before comparison to the marker data, the echo correction was applied to all virtual models and echo measurements. An example of a virtual model before and after echo correction is shown in Figure 5.6. Here, the model before correction is in blue and after correction is in black. It can be seen that the error from the *in vitro* setup caused the measured echo to represent the valve as smaller than it actually was.

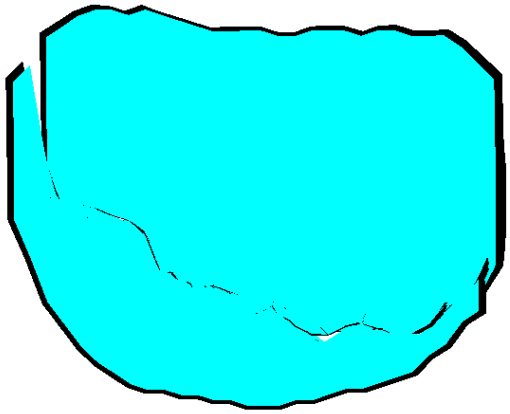


Figure 5.6 Overlay of uncorrected (blue) and corrected (black) models using the scheme described in section 4.3.8. Atrial chamber 3 was used during the acquisition of the data set used to create these virtual models.

5.4 Dynamic Valve Results

A total of three time points from two normal valves and two disease conditions were investigated. Each valve was studied during end diastole (closing), peak systole

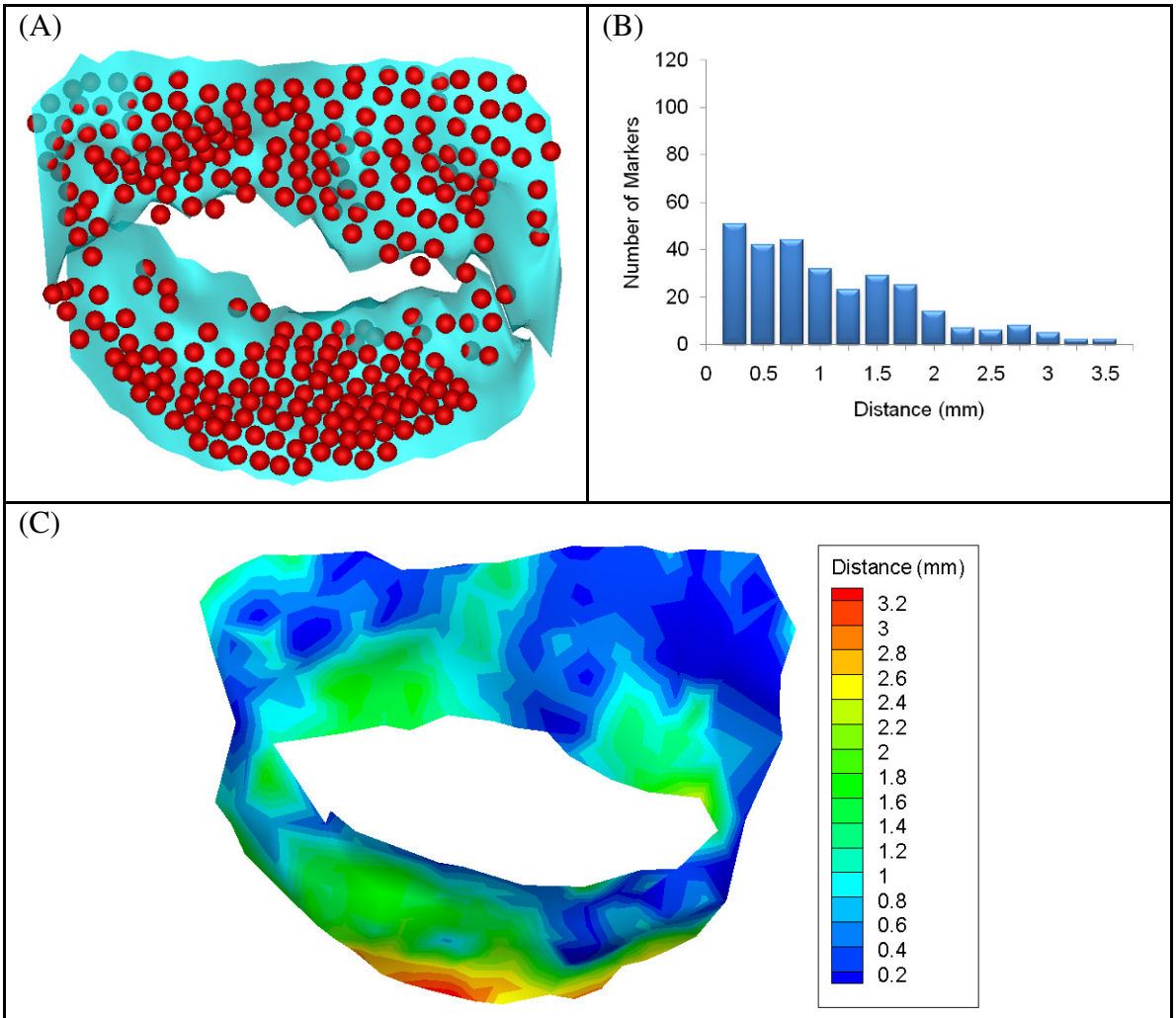
(closed), and early diastole (opening). Marker data for each condition was reconstructed into 3D coordinates using a direct linear transformation as described in section 4.3.6. The minimum distance between each marker point and the corresponding virtual model was calculated. The mean, standard deviation, and RMS distance for each condition were also calculated. Every condition was evaluated after transformation to the cube basis and best fit alignment of the marker data and the virtual model.

The results for each condition are presented as a single figure divided into four sections. Section A shows an image of the marker data (red spheres) and the virtual model (blue surface). Section B contains the distribution of distances measured between each marker and the virtual model. Section C shows the distance at each marker mapped onto a surface created between the marker vertices. The mean, standard deviation, and RMS distance values are listed at the bottom of each figure. The following is an outline of the order of the results and corresponding figures.

1) Normal Valve
    a. Closing Valve 1 after Cube Transformation (Figure 5.7)
    b. Closing Valve 1 after Best Fit (Figure 5.8)
    c. Peak Systolic Valve 1 after Cube Transformation (Figure 5.9)
    d. Peak Systolic Valve 1 after Best Fit (Figure 5.10)
    e. Opening Valve 1 after Cube Transformation (Figure 5.11)
    f. Opening Valve 1 after Best Fit (Figure 5.12)
    g. Closing Valve 2 after Cube Transformation (Figure 5.13)
    h. Closing Valve 2 after Best Fit (Figure 5.14)
    i. Peak Systolic Valve 2 after Cube Transformation (Figure 5.15)
    j. Peak Systolic Valve 2 after Best Fit (Figure 5.16)
    k. Opening Valve 2 after Cube Transformation (Figure 5.17)
    l. Opening Valve 2 after Best Fit (Figure 5.18)

2) Valve with Flail Leaflet
    a. Closing after Cube Transformation (Figure 5.19)
    b. Closing after Best Fit (Figure 5.20)
    c. Peak Systolic after Cube Transformation (Figure 5.21)
    d. Peak Systolic after Best Fit (Figure 5.22)
    e. Opening after Cube Transformation (Figure 5.23)
    f. Opening after Best Fit (Figure 5.24)
3) Valve with Billowing Leaflet
    a. Closing after Cube Transformation (Figure 5.25)
    b. Closing after Best Fit (Figure 5.26)
    c. Peak Systolic after Cube Transformation (Figure 5.27)
    d. Peak Systolic after Best Fit (Figure 5.28)
    e. Opening after Cube Transformation (Figure 5.29)
    f. Opening after Best Fit (Figure 5.30)
4) Summary Tables
    a. Summary of Belly and Commissure Errors for Dynamic Cases (Table 5.3)
    b. Summary of Total and RMS Errors for Dynamic Cases (Table 5.4)

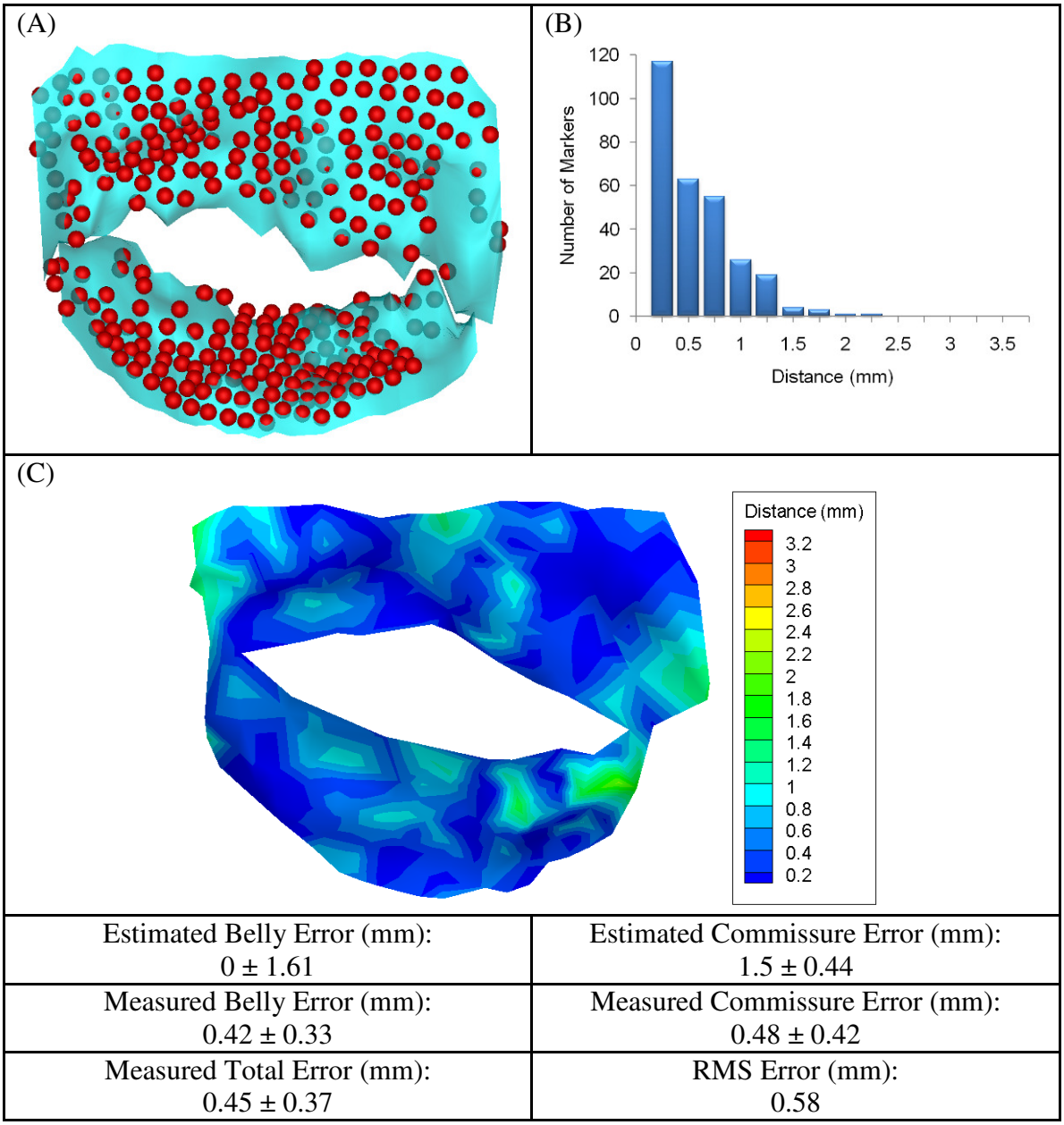
Closing Valve 1 after Cube Transformation



Estimated Belly Error (mm): 0.63 ± 1.61	Estimated Commissure Error (mm): 2.13 ± 0.44
Measured Belly Error (mm): 1.06 ± 0.80	Measured Commissure Error (mm): 0.91 ± 0.70
Measured Total Error (mm): 0.99 ± 0.75	RMS Error (mm): 1.24

Figure 5.7 Results for normal valve 1 after cube transformation while closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

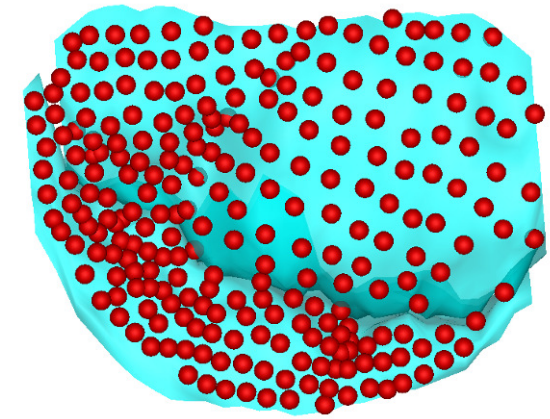
Closing Valve 1 after Best Fit

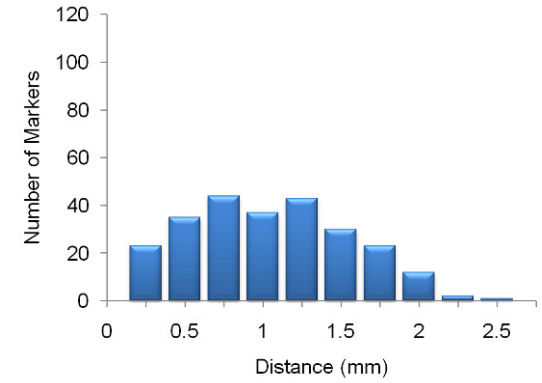


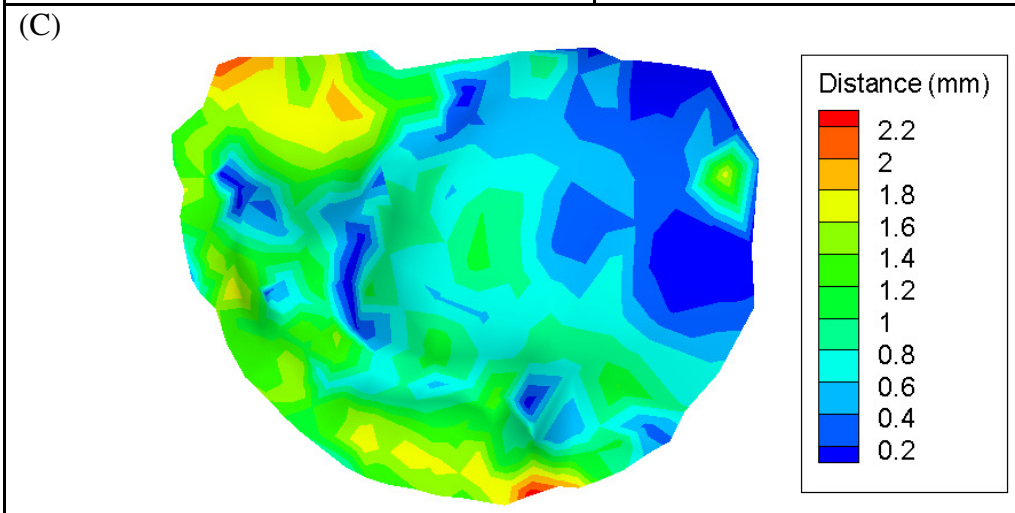
Estimated Belly Error (mm): 0 ± 1.61	Estimated Commissure Error (mm): 1.5 ± 0.44
Measured Belly Error (mm): 0.42 ± 0.33	Measured Commissure Error (mm): 0.48 ± 0.42
Measured Total Error (mm): 0.45 ± 0.37	RMS Error (mm): 0.58

Figure 5.8 Results for normal valve 1 after best fit while closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Peak Systolic Valve 1 after Cube Transformation



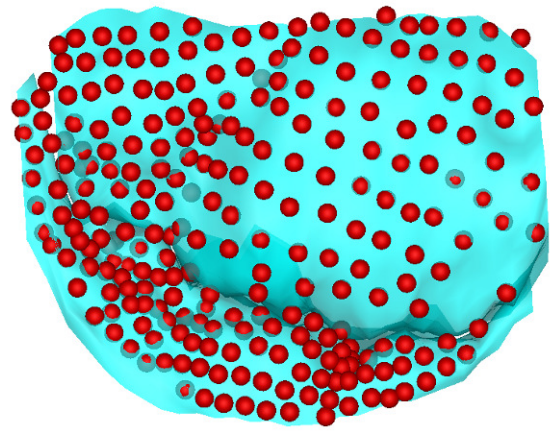


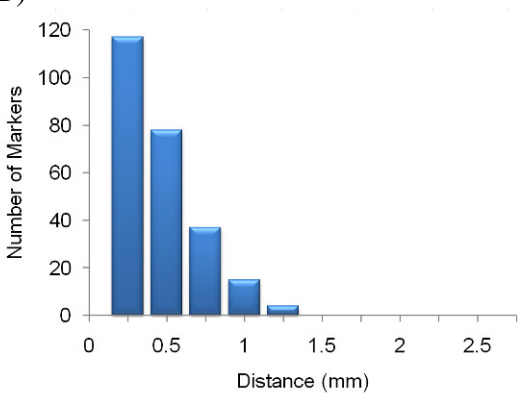


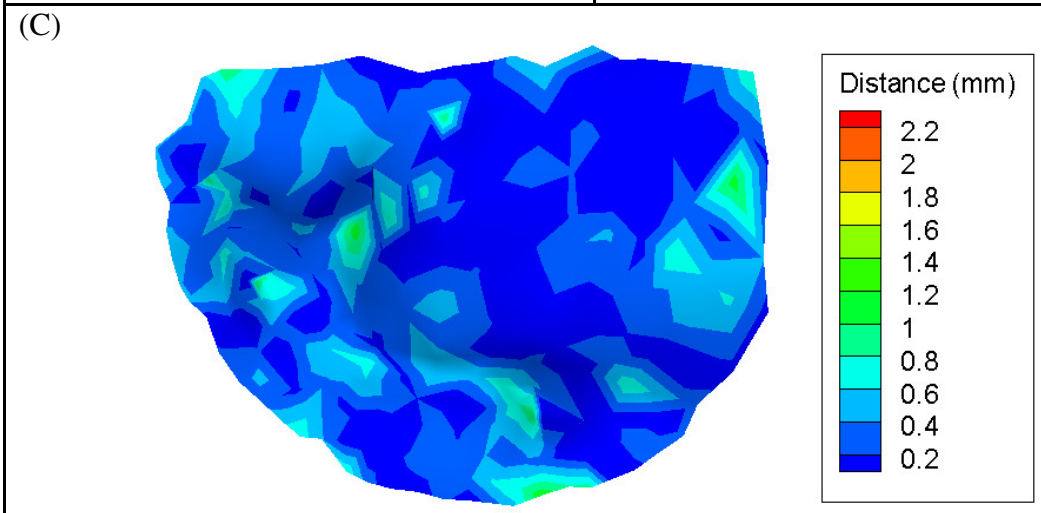
Estimated Belly Error (mm): 0.63 ± 0.25	Estimated Commissure Error (mm): 2.13 ± 0.25
Measured Belly Error (mm): 0.93 ± 0.56	Measured Commissure Error (mm): 0.92 ± 0.46
Measured Total Error (mm): 0.93 ± 0.51	RMS Error (mm): 1.06

Figure 5.9 Results for normal valve 1 after cube transformation at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Peak Systolic Valve 1 after Best Fit



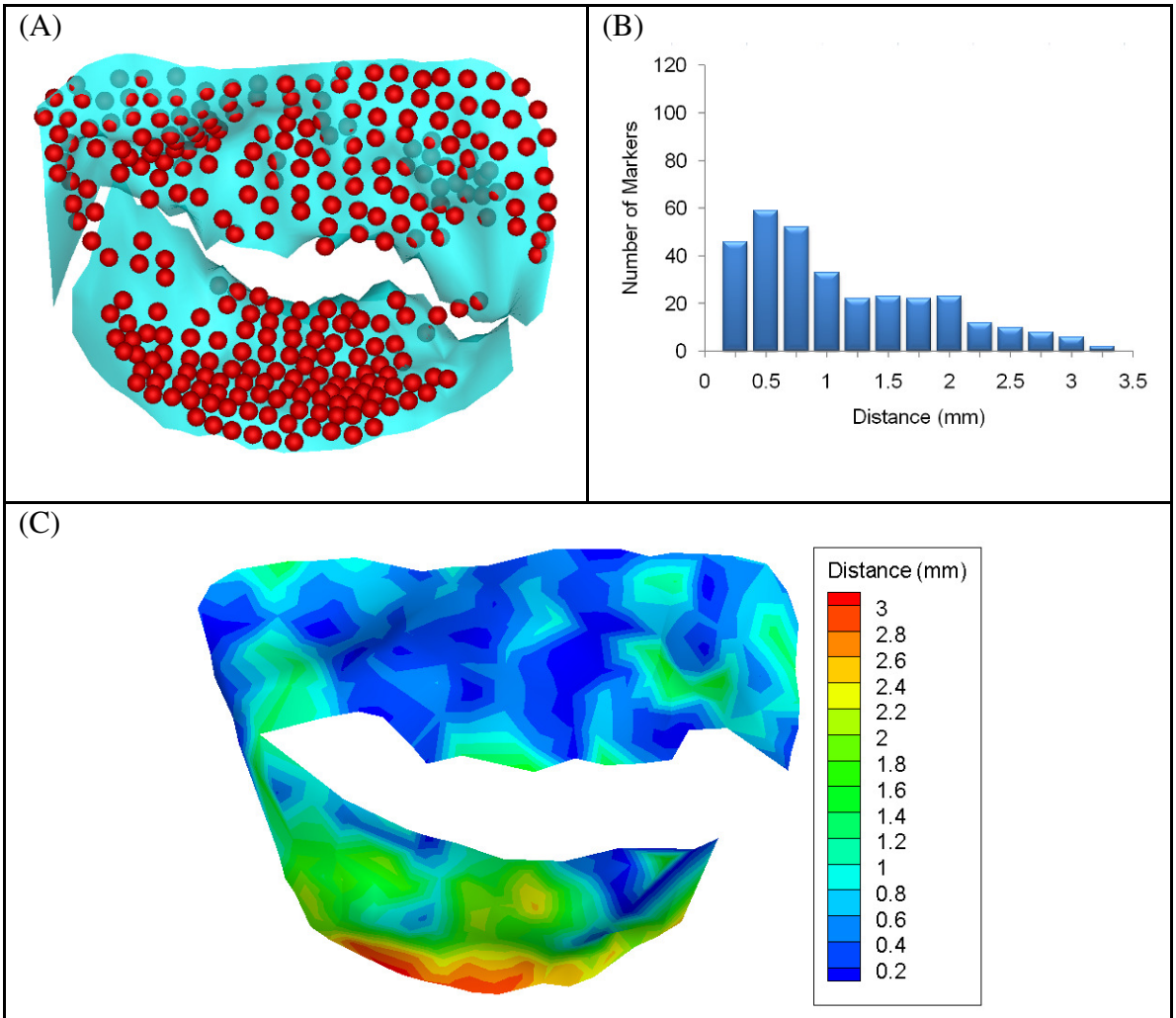




Estimated Belly Error (mm): 0 ± 0.25	Estimated Commissure Error (mm): 1.5 ± 0.25
Measured Belly Error (mm): 0.35 ± 0.23	Measured Commissure Error (mm): 0.31 ± 0.28
Measured Total Error (mm): 0.33 ± 0.25	RMS Error (mm): 0.41

Figure 5.10 Results for normal valve 1 after best fit at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

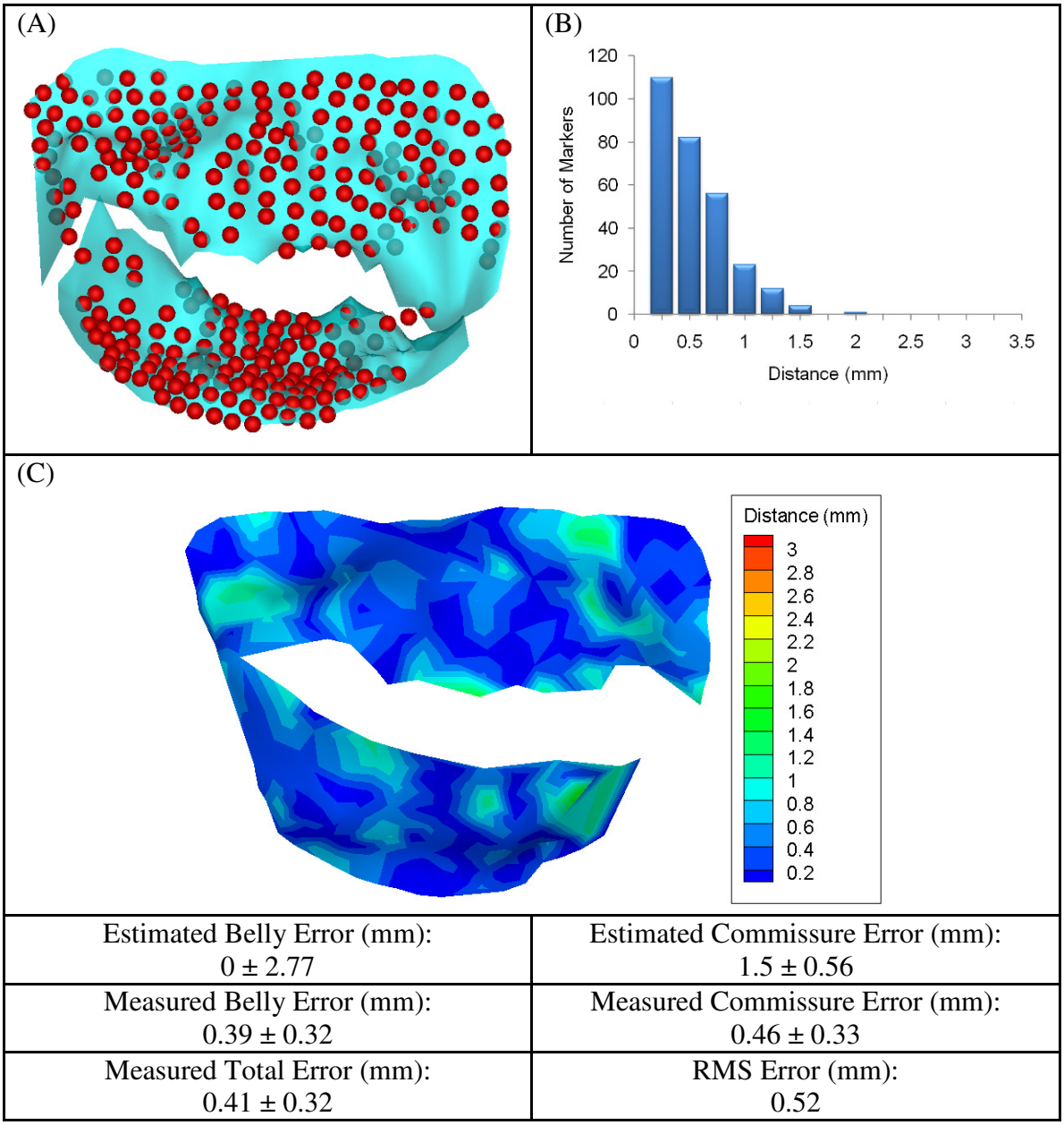
Opening Valve 1 after Cube Transformation



Estimated Belly Error (mm): 0.63 ± 2.77	Estimated Commissure Error (mm): 2.13 ± 0.56
Measured Belly Error (mm): 1.09 ± 0.83	Measured Commissure Error (mm): 0.85 ± 0.61
Measured Total Error (mm): 0.99 ± 0.76	RMS Error (mm): 1.25

Figure 5.11 Results for normal valve 1 after cube transformation during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

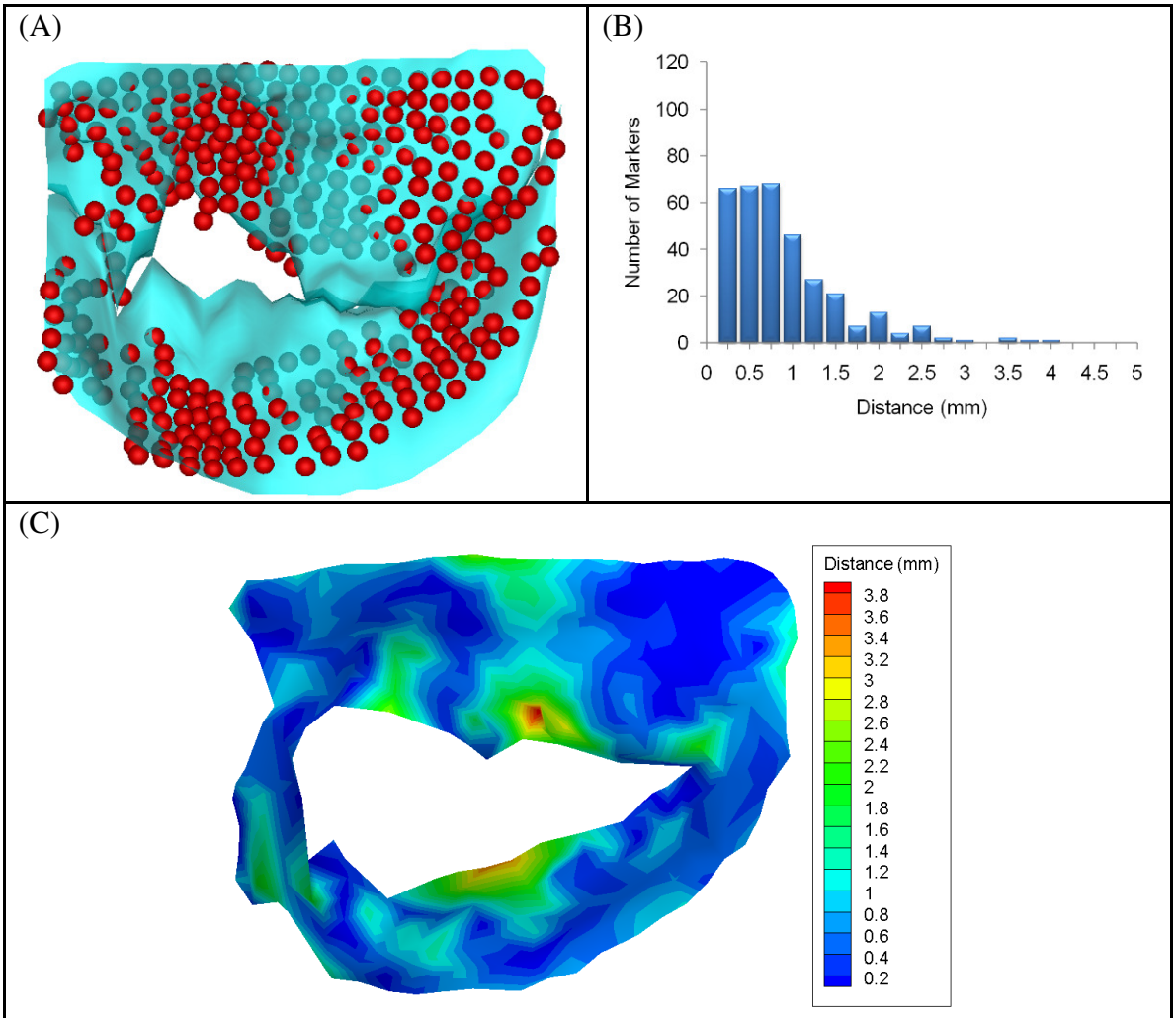
Opening Valve 1 after Best Fit



Estimated Belly Error (mm): 0 ± 2.77	Estimated Commissure Error (mm): 1.5 ± 0.56
Measured Belly Error (mm): 0.39 ± 0.32	Measured Commissure Error (mm): 0.46 ± 0.33
Measured Total Error (mm): 0.41 ± 0.32	RMS Error (mm): 0.52

Figure 5.12 Results for normal valve 1 after best fit during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

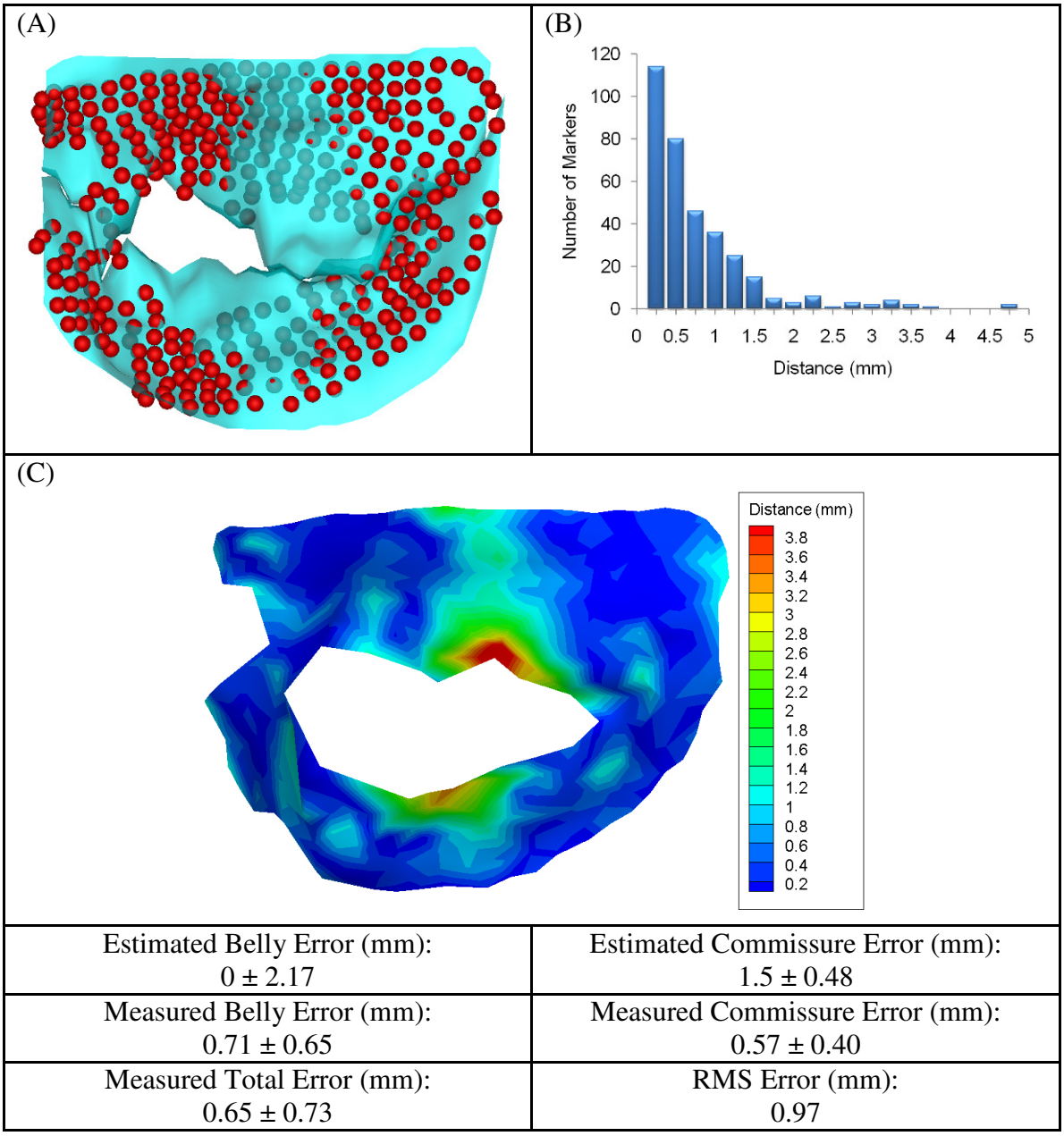
Closing Valve 2 after Cube Transformation



Estimated Belly Error (mm): 0.63 ± 2.17	Estimated Commissure Error (mm): 2.13 ± 0.48
Measured Belly Error (mm): 0.85 ± 0.74	Measured Commissure Error (mm): 0.63 ± 0.47
Measured Total Error (mm): 0.77 ± 0.66	RMS Error (mm): 1.01

Figure 5.13 Results for normal valve 2 after cube transformation during closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

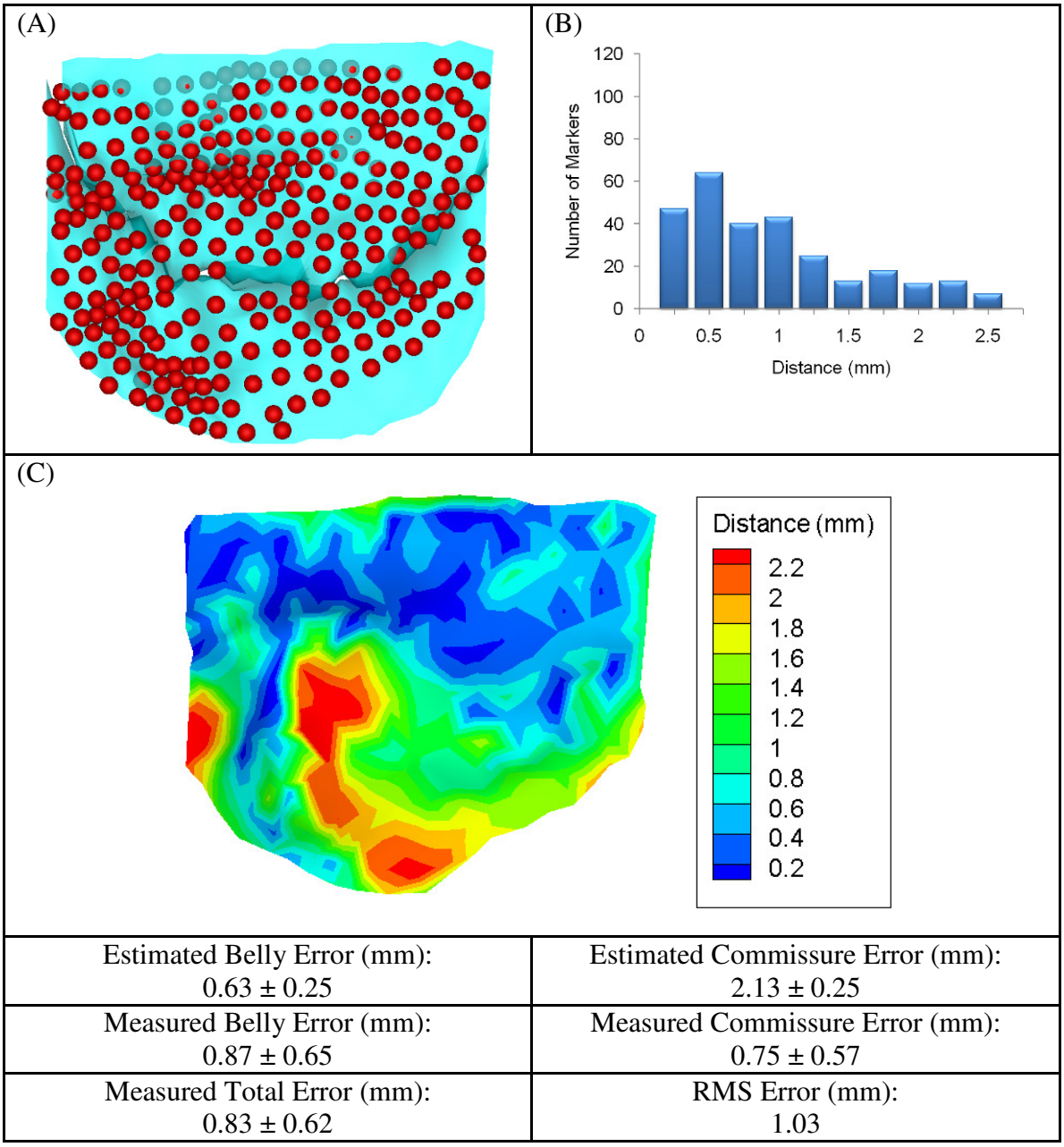
Closing Valve 2 after Best Fit



Estimated Belly Error (mm): 0 ± 2.17	Estimated Commissure Error (mm): 1.5 ± 0.48
Measured Belly Error (mm): 0.71 ± 0.65	Measured Commissure Error (mm): 0.57 ± 0.40
Measured Total Error (mm): 0.65 ± 0.73	RMS Error (mm): 0.97

Figure 5.14 Results for normal valve 2 after best fit during closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

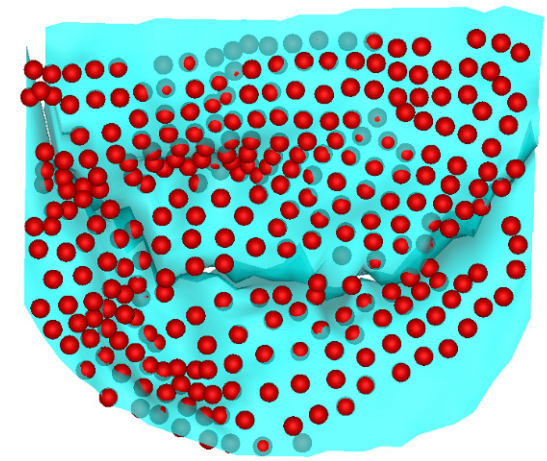
Peak Systolic Valve 2 after Cube Transformation

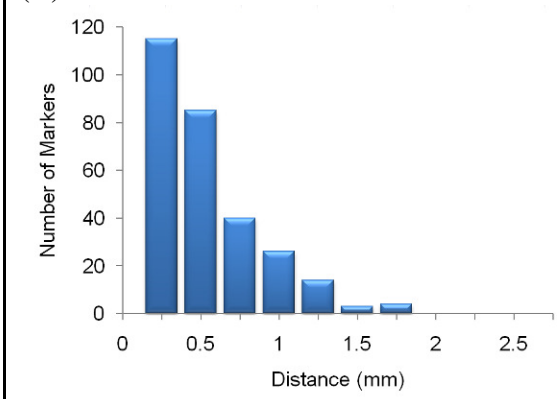


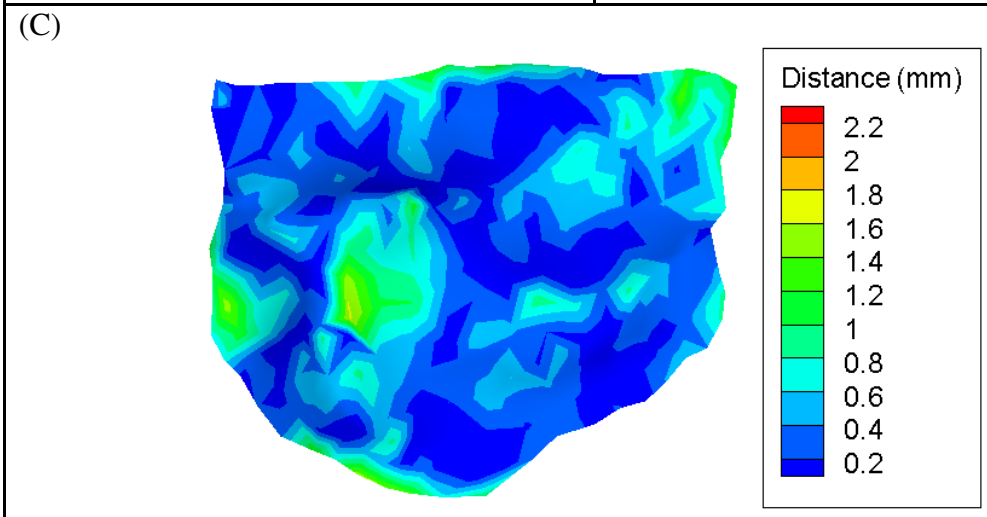
Estimated Belly Error (mm): 0.63 ± 0.25	Estimated Commissure Error (mm): 2.13 ± 0.25
Measured Belly Error (mm): 0.87 ± 0.65	Measured Commissure Error (mm): 0.75 ± 0.57
Measured Total Error (mm): 0.83 ± 0.62	RMS Error (mm): 1.03

Figure 5.15 Results for normal valve 2 after cube transformation at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Peak Systolic Valve 2 after Best Fit



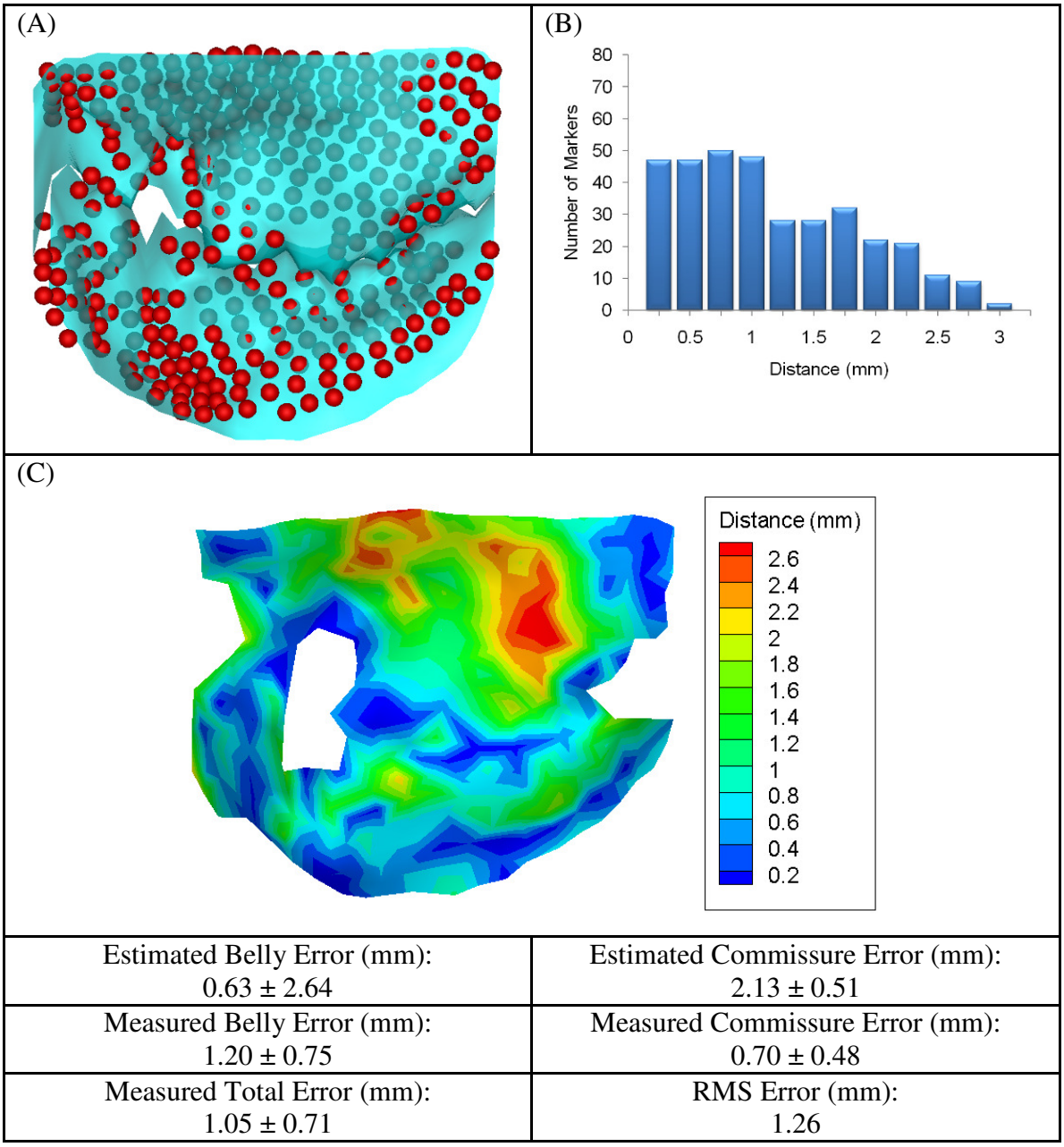




Estimated Belly Error (mm): 0 ± 0.25	Estimated Commissure Error (mm): 1.5 ± 0.25
Measured Belly Error (mm): 0.41 ± 0.34	Measured Commissure Error (mm): 0.43 ± 0.33
Measured Total Error (mm): 0.42 ± 0.34	RMS Error (mm): 0.54

Figure 5.16 Results for normal valve 2 after best fit at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Opening Valve 2 after Cube Transformation

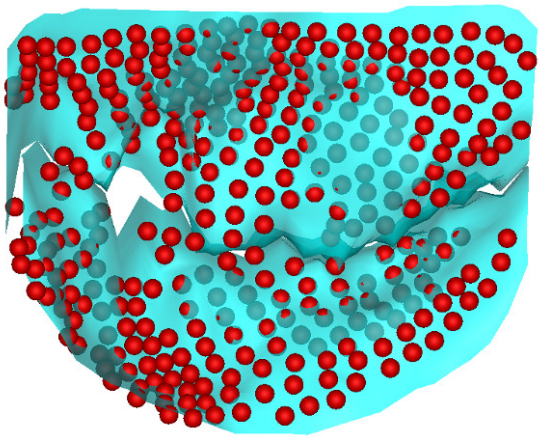


Estimated Belly Error (mm): 0.63 ± 2.64	Estimated Commissure Error (mm): 2.13 ± 0.51
Measured Belly Error (mm): 1.20 ± 0.75	Measured Commissure Error (mm): 0.70 ± 0.48
Measured Total Error (mm): 1.05 ± 0.71	RMS Error (mm): 1.26

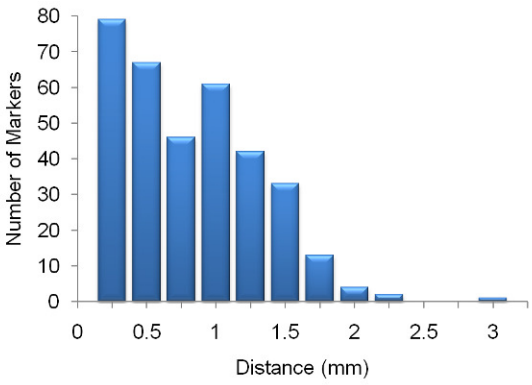
Figure 5.17 Results for normal valve 2 after cube transformation during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Opening Valve 2 after Best Fit

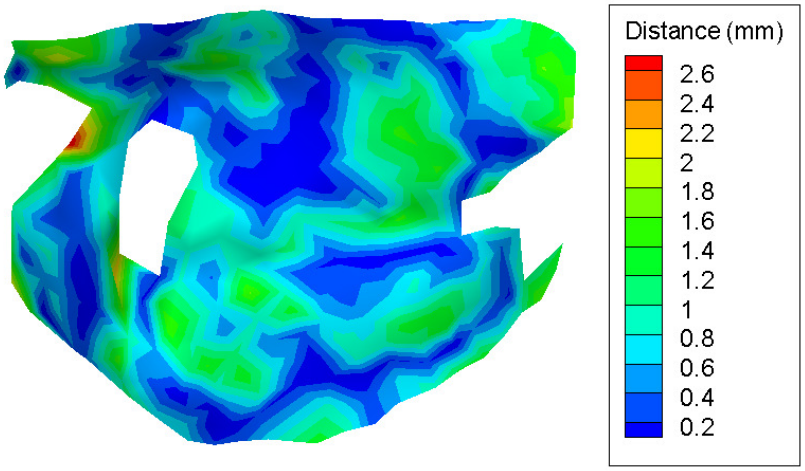
(A)



(B)



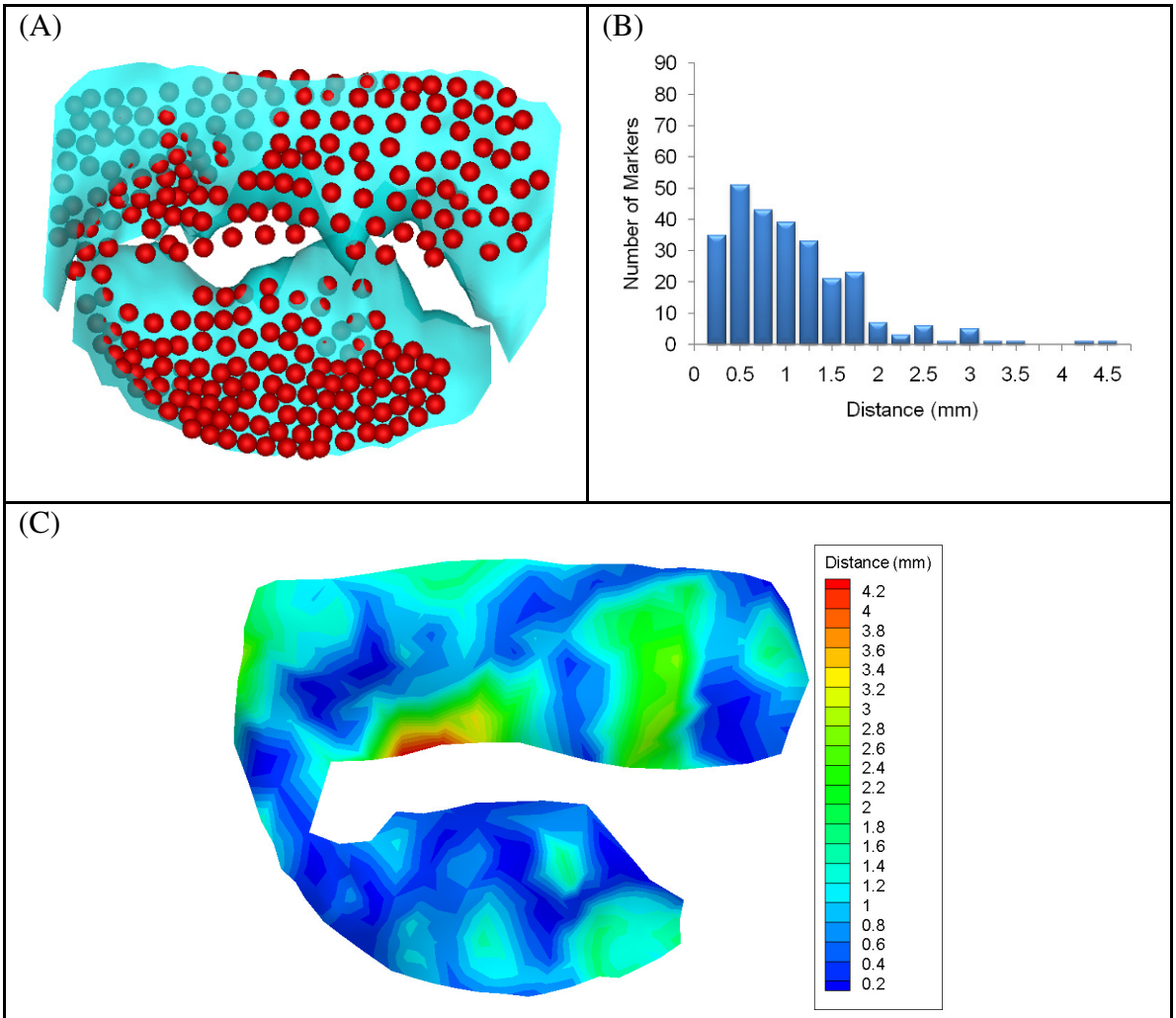
(C)



Estimated Belly Error (mm): 0 ± 2.64	Estimated Commissure Error (mm): 1.5 ± 0.53
Measured Belly Error (mm): 0.63 ± 0.42	Measured Commissure Error (mm): 0.85 ± 0.57
Measured Total Error (mm): 0.71 ± 0.49	RMS Error (mm): 0.86

Figure 5.18 Results for normal valve 2 after best fit during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

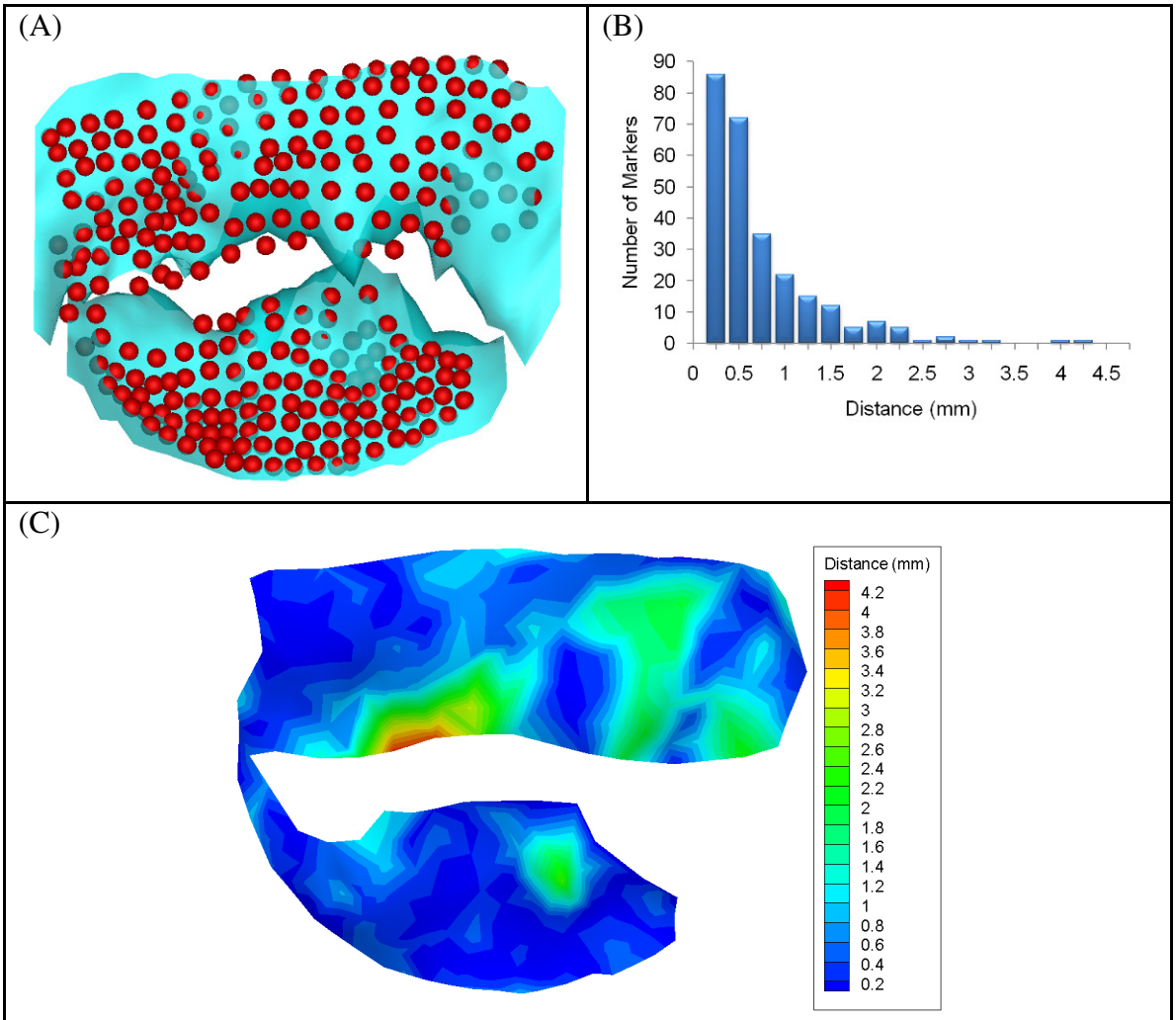
Closing Valve with Flail Leaflet after Cube Transformation



Estimated Belly Error (mm): 0.63 ± 3.55	Estimated Commissure Error (mm): 2.13 ± 0.71
Measured Belly Error (mm): 0.87 ± 0.66	Measured Commissure Error (mm): 0.98 ± 0.75
Measured Total Error (mm): 0.93 ± 0.71	RMS Error (mm): 1.17

Figure 5.19 Results for the valve with a flail leaflet after cube transformation during closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

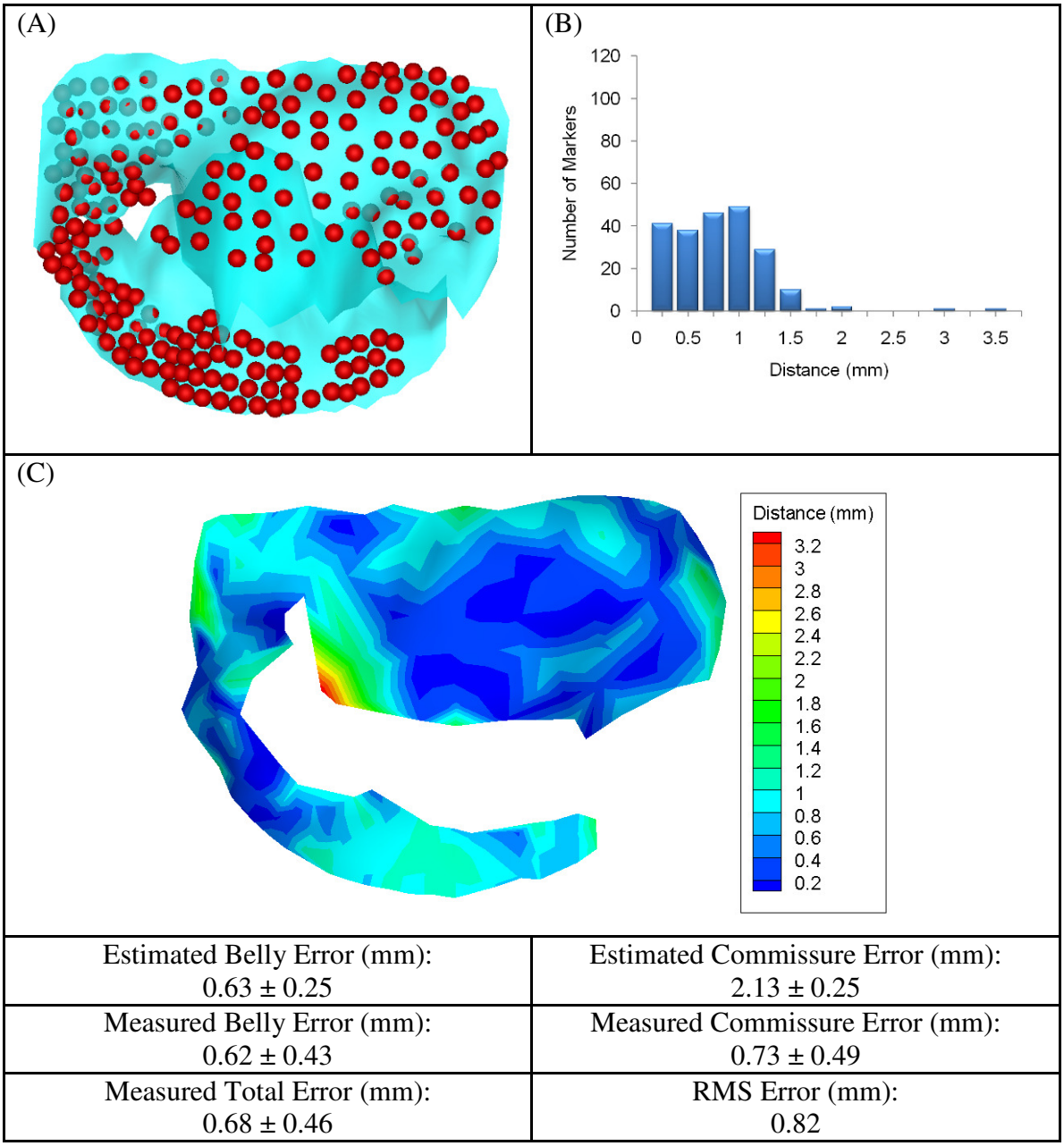
Closing Valve with Flail Leaflet after Best Fit



Estimated Belly Error (mm): 0 ± 3.55	Estimated Commissure Error (mm): 1.5 ± 0.71
Measured Belly Error (mm): 0.46 ± 0.43	Measured Commissure Error (mm): 0.69 ± 0.72
Measured Total Error (mm): 0.62 ± 0.65	RMS Error (mm): 0.89

Figure 5.20 Results for the valve with a flail leaflet after best fit during closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

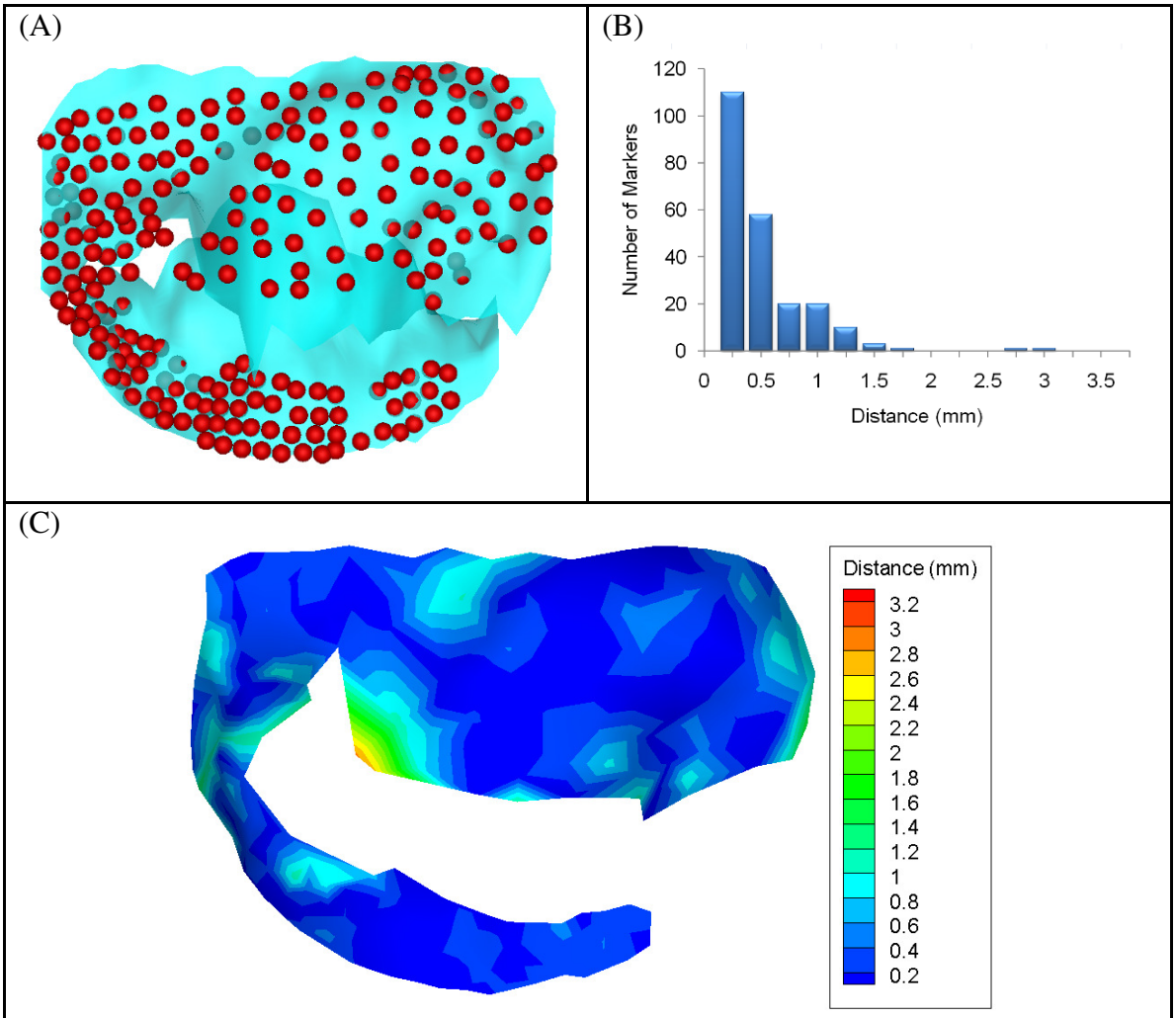
Peak Systolic Valve with Flail Leaflet after Cube Transformation



Estimated Belly Error (mm): 0.63 ± 0.25	Estimated Commissure Error (mm): 2.13 ± 0.25
Measured Belly Error (mm): 0.62 ± 0.43	Measured Commissure Error (mm): 0.73 ± 0.49
Measured Total Error (mm): 0.68 ± 0.46	RMS Error (mm): 0.82

Figure 5.21 Results for the valve with a flail leaflet after cube transformation at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

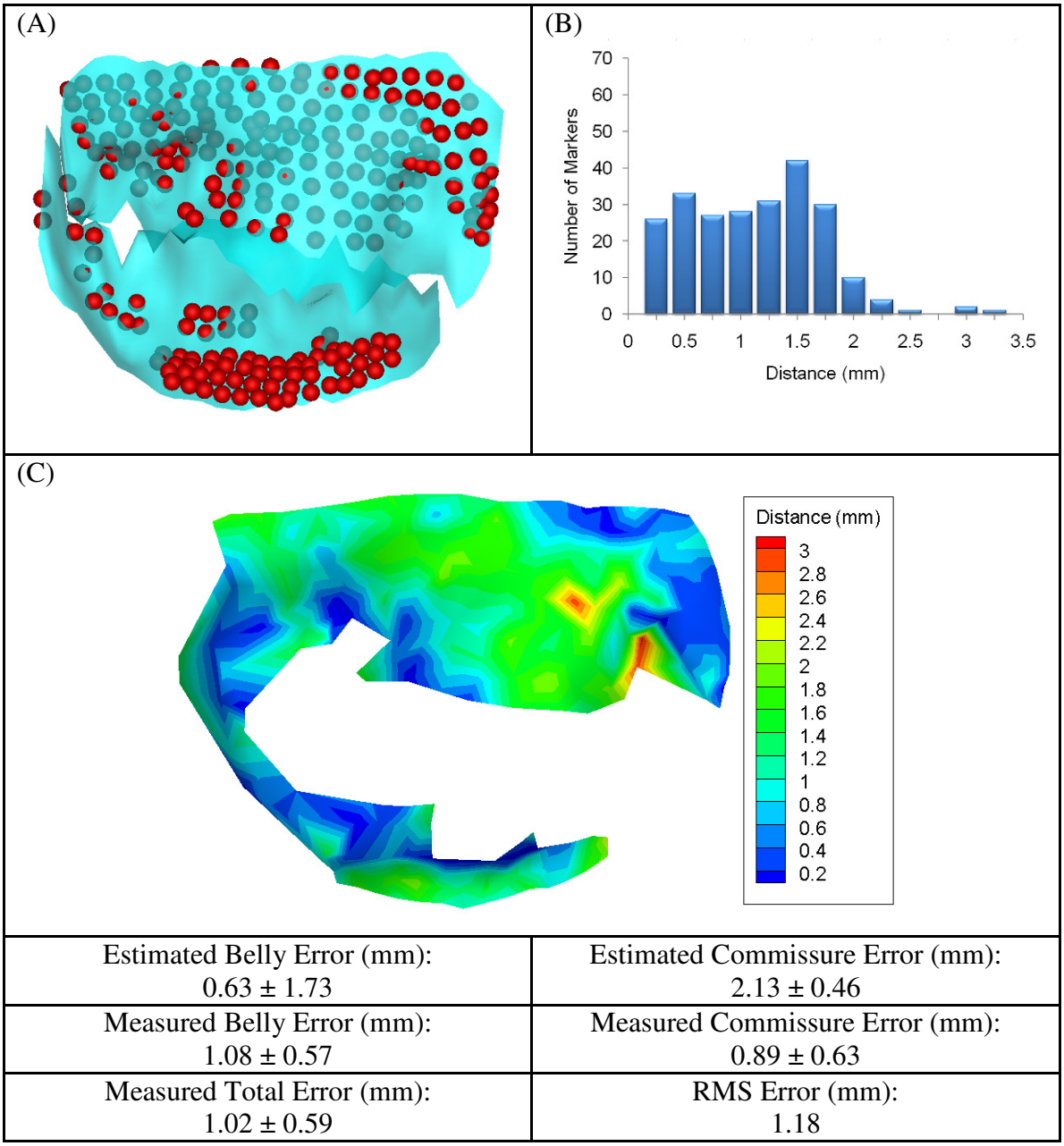
Peak Systolic Valve with Flail Leaflet after Best Fit



Estimated Belly Error (mm): 0 ± 0.25	Estimated Commissure Error (mm): 1.5 ± 0.25
Measured Belly Error (mm): 0.45 ± 0.36	Measured Commissure Error (mm): 0.32 ± 0.42
Measured Total Error (mm): 0.38 ± 0.40	RMS Error (mm): 0.55

Figure 5.22 Results for the valve with a flail leaflet after best fit at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

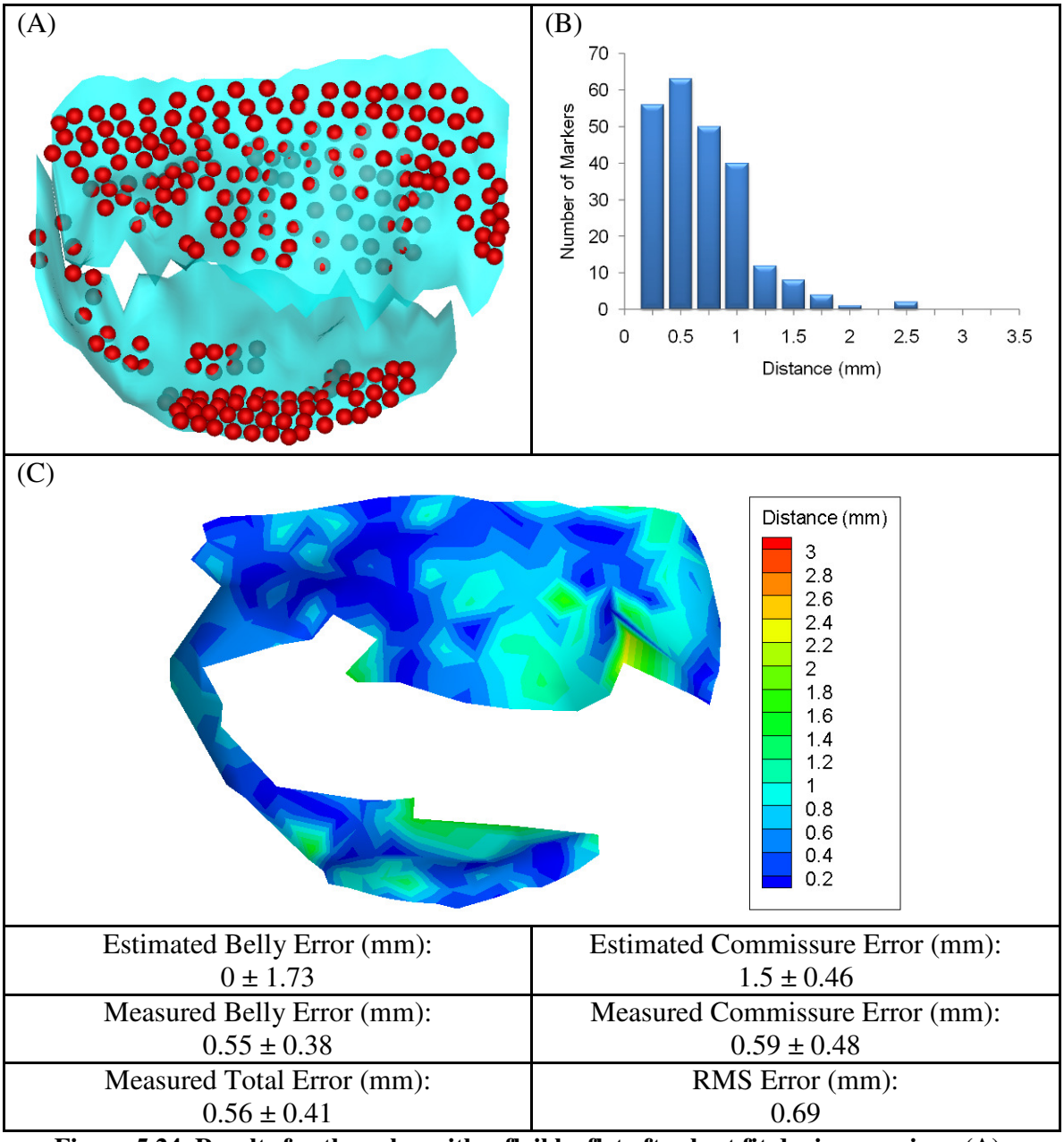
Opening Valve with Flail Leaflet after Cube Transformation



Estimated Belly Error (mm): 0.63 ± 1.73	Estimated Commissure Error (mm): 2.13 ± 0.46
Measured Belly Error (mm): 1.08 ± 0.57	Measured Commissure Error (mm): 0.89 ± 0.63
Measured Total Error (mm): 1.02 ± 0.59	RMS Error (mm): 1.18

Figure 5.23 Results for the valve with a flail leaflet after cube transformation during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

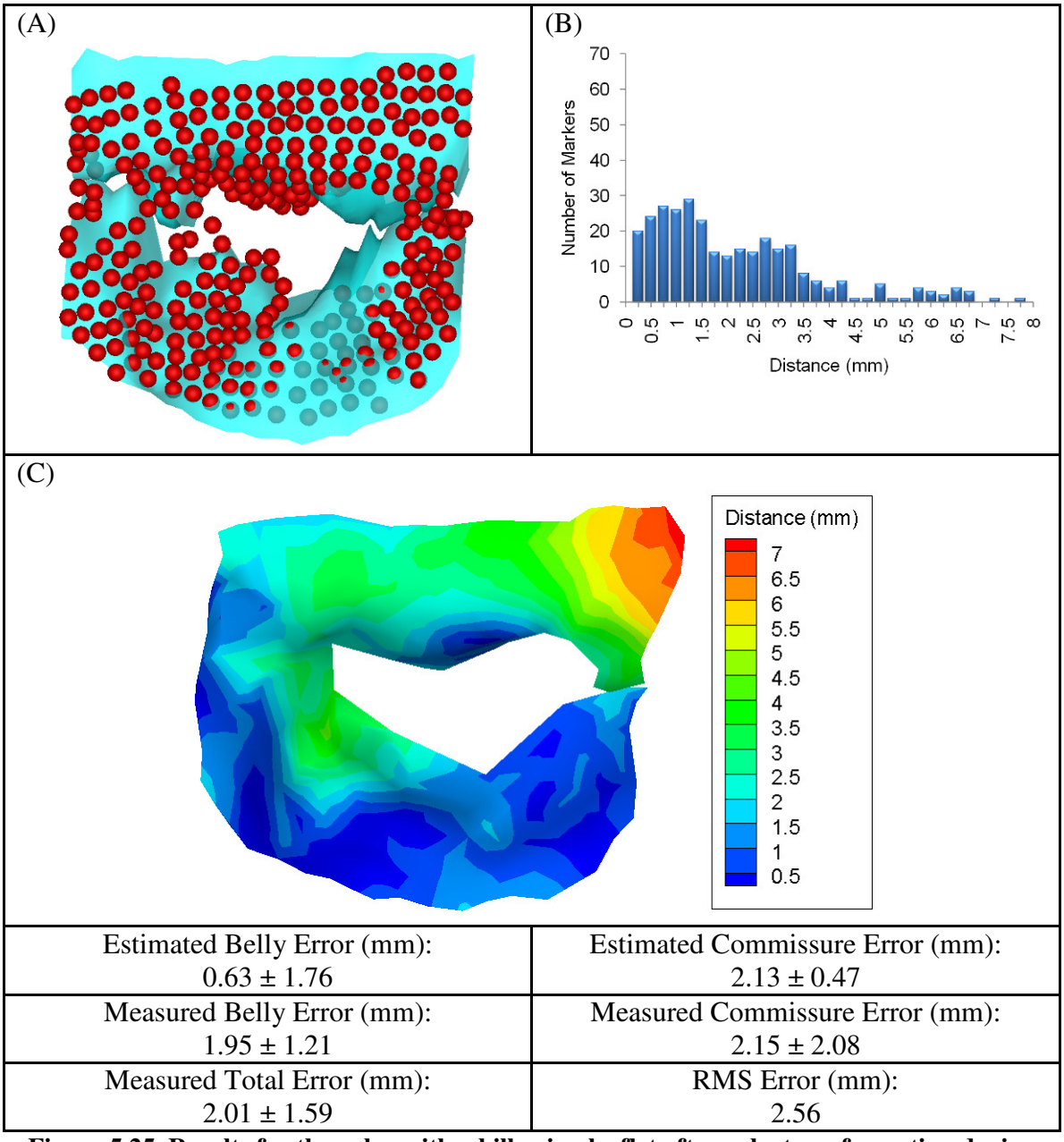
Opening Valve with Flail Leaflet after Best Fit



Estimated Belly Error (mm): 0 ± 1.73	Estimated Commissure Error (mm): 1.5 ± 0.46
Measured Belly Error (mm): 0.55 ± 0.38	Measured Commissure Error (mm): 0.59 ± 0.48
Measured Total Error (mm): 0.56 ± 0.41	RMS Error (mm): 0.69

Figure 5.24 Results for the valve with a flail leaflet after best fit during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

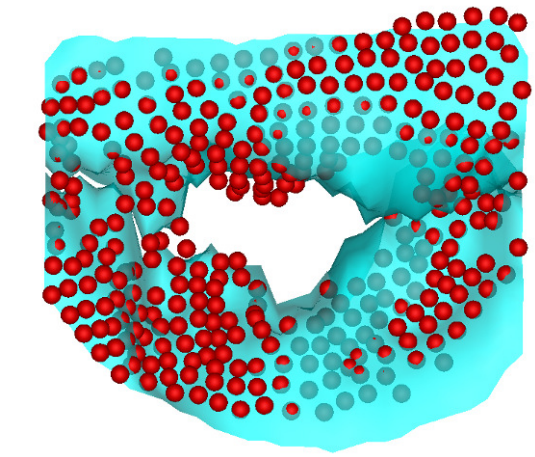
Closing Valve with Billowing Leaflet after Cube Transformation

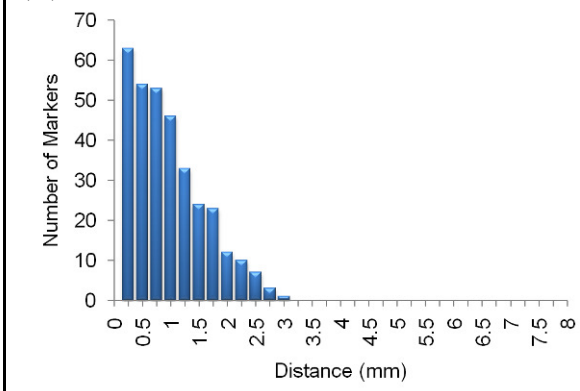


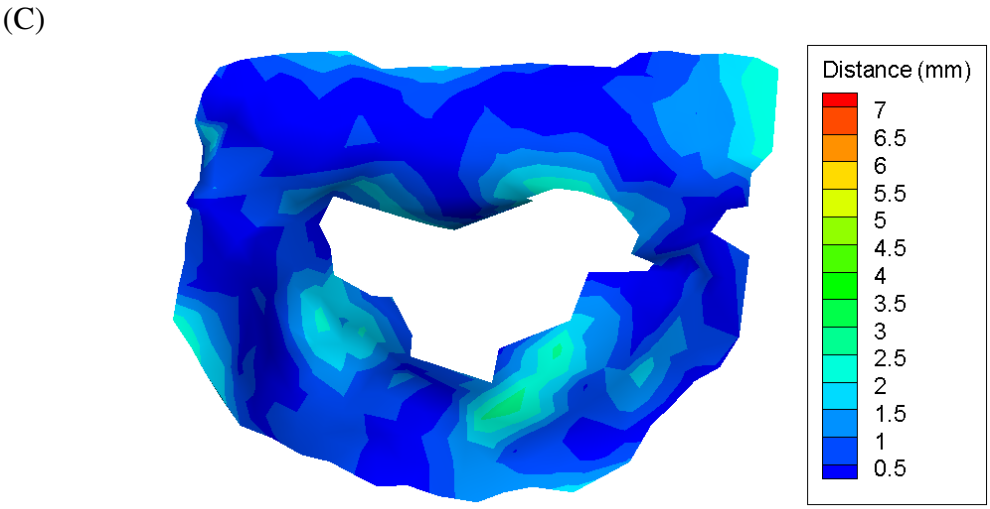
Estimated Belly Error (mm): 0.63 ± 1.76	Estimated Commissure Error (mm): 2.13 ± 0.47
Measured Belly Error (mm): 1.95 ± 1.21	Measured Commissure Error (mm): 2.15 ± 2.08
Measured Total Error (mm): 2.01 ± 1.59	RMS Error (mm): 2.56

Figure 5.25 Results for the valve with a billowing leaflet after cube transformation during closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Closing Valve with Billowing Leaflet after Best Fit



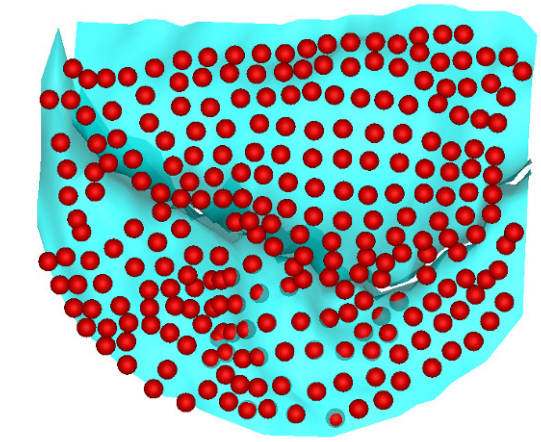


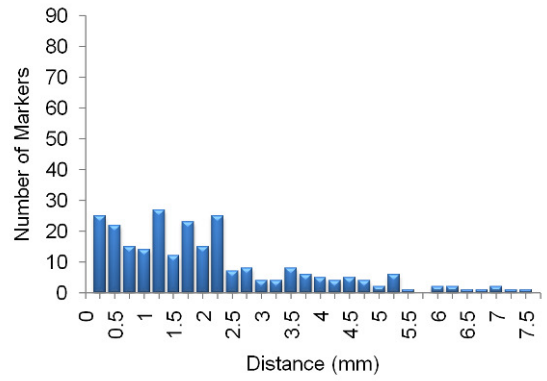


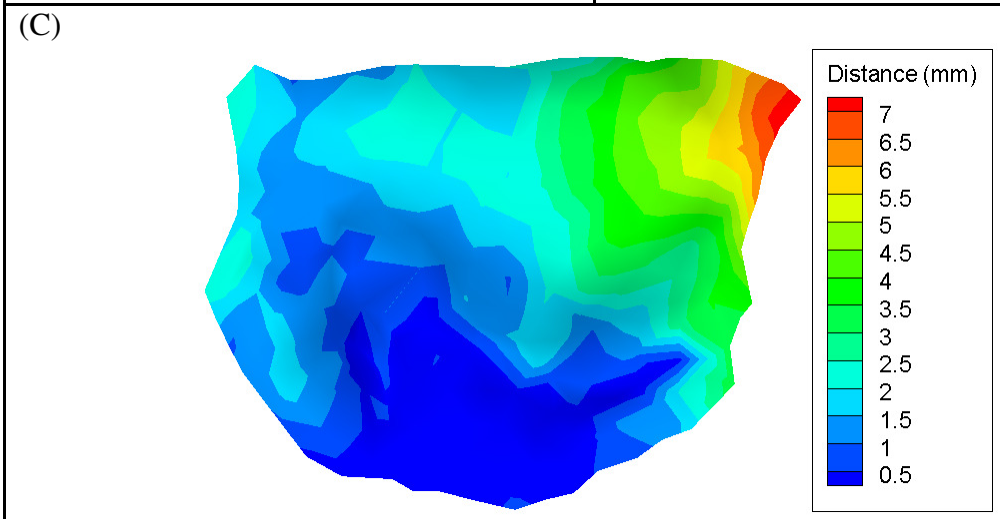
Estimated Belly Error (mm): 0 ± 1.76	Estimated Commissure Error (mm): 1.5 ± 0.47
Measured Belly Error (mm): 0.88 ± 0.63	Measured Commissure Error (mm): 0.78 ± 0.63
Measured Total Error (mm): 0.85 ± 0.63	RMS Error (mm): 1.06

Figure 5.26 Results for the valve with a billowing leaflet after best fit during closing. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Peak Systolic Valve with Billowing Leaflet after Cube Transformation



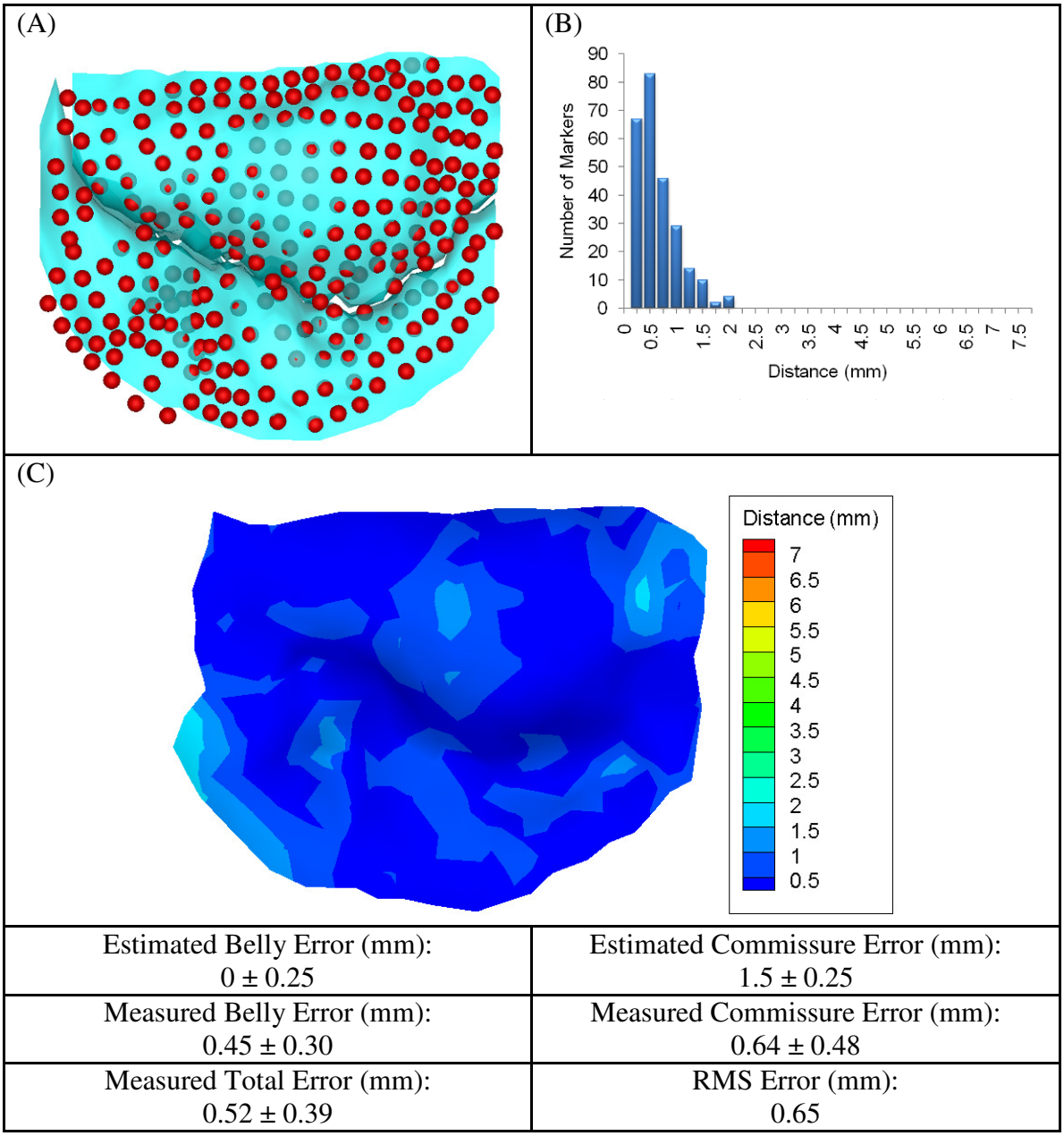




Estimated Belly Error (mm): 0.63 ± 0.25	Estimated Commissure Error (mm): 2.13 ± 0.25
Measured Belly Error (mm): 1.51 ± 1.20	Measured Commissure Error (mm): 2.90 ± 1.92
Measured Total Error (mm): 1.97 ± 1.61	RMS Error (mm): 2.54

Figure 5.27 Results for the valve with a billowing leaflet after cube transformation at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

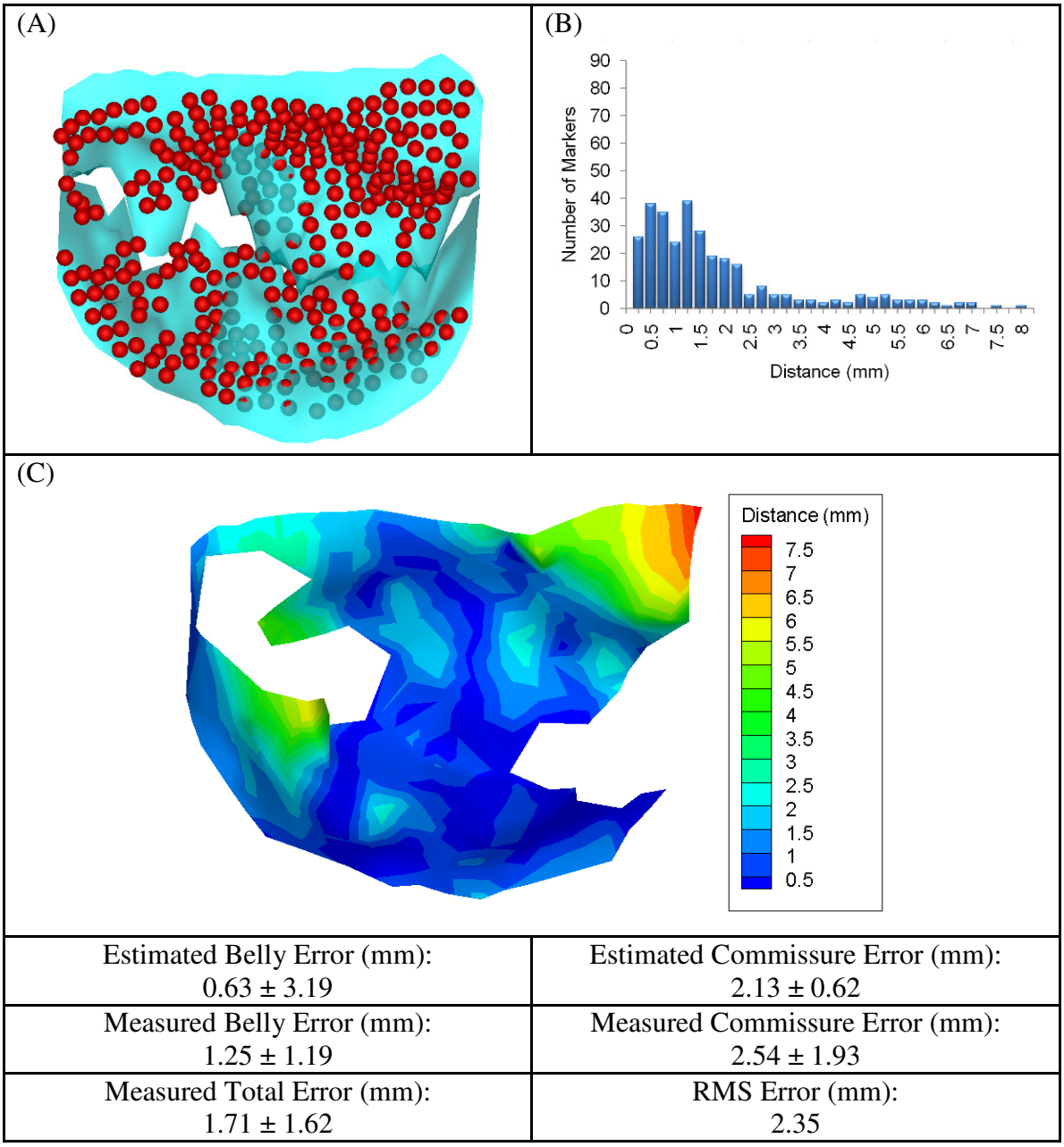
Peak Systolic Valve with Billowing Leaflet after Best Fit



Estimated Belly Error (mm): 0 ± 0.25	Estimated Commissure Error (mm): 1.5 ± 0.25
Measured Belly Error (mm): 0.45 ± 0.30	Measured Commissure Error (mm): 0.64 ± 0.48
Measured Total Error (mm): 0.52 ± 0.39	RMS Error (mm): 0.65

Figure 5.28 Results for the valve with a billowing leaflet after best fit at peak systole. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

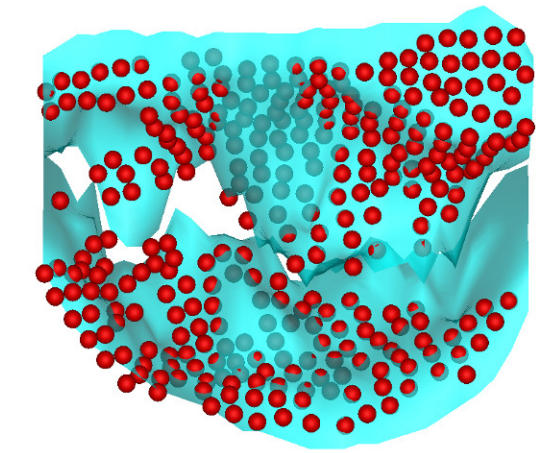
Opening Valve with Billowing Leaflet after Cube Transformation

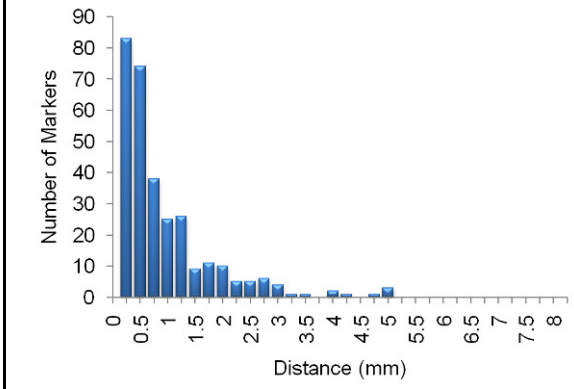


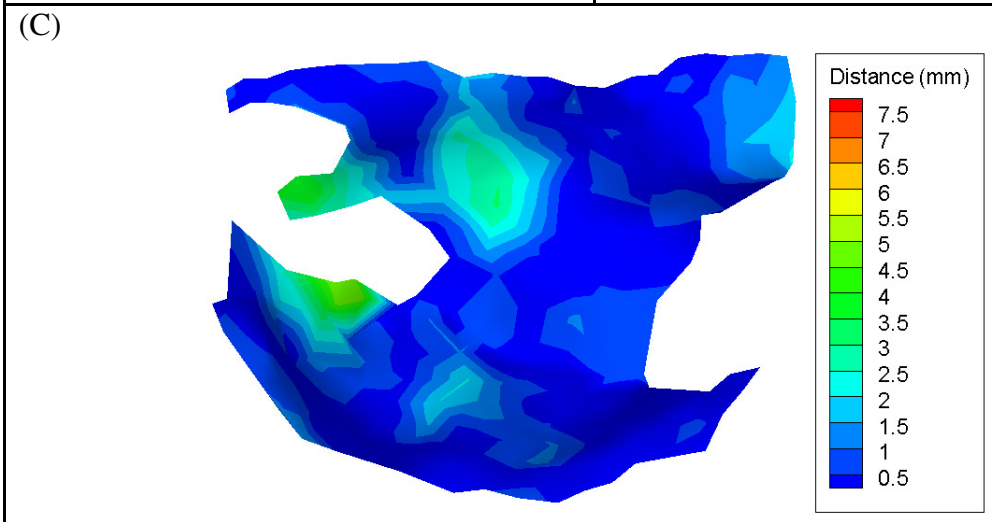
Estimated Belly Error (mm): 0.63 ± 3.19	Estimated Commissure Error (mm): 2.13 ± 0.62
Measured Belly Error (mm): 1.25 ± 1.19	Measured Commissure Error (mm): 2.54 ± 1.93
Measured Total Error (mm): 1.71 ± 1.62	RMS Error (mm): 2.35

Figure 5.29 Results for the valve with a billowing leaflet after cube transformation during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Opening Valve with Billowing Leaflet after Best Fit







Estimated Belly Error (mm): 0 ± 3.19	Estimated Commissure Error (mm): 1.5 ± 0.62
Measured Belly Error (mm): 0.79 ± 0.89	Measured Commissure Error (mm): 0.86 ± 0.89
Measured Total Error (mm): 0.82 ± 0.89	RMS Error (mm): 1.21

Figure 5.30 Results for the valve with a billow leaflet after best fit during opening. (A): Marker data (red spheres) and virtual model after cube transformation (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Table 5.3 Summary of Belly and Commissure Errors for Dynamic Cases

	Estimated Belly Error (mm)	Estimated Comm Error (mm)	Measured Belly Error (mm)	Measured Comm Error (mm)
Normal 1 Closing Cube	0.63 ± 1.61	2.13 ± 0.44	1.06 ± 0.80	0.91 ± 0.70
Normal 1 Peak Systolic Cube	0.63 ± 0.25	2.13 ± 0.25	0.93 ± 0.56	0.92 ± 0.46
Normal 1 Opening Cube	0.63 ± 2.77	2.13 ± 0.56	1.09 ± 0.83	0.85 ± 0.61
Normal 1 Closing Best	0 ± 1.61	1.5 ± 0.44	0.42 ± 0.33	0.48 ± 0.42
Normal 1 Peak Systolic Best	0 ± 0.25	1.5 ± 0.25	0.35 ± 0.23	0.31 ± 0.28
Normal 1 Opening Best	0 ± 2.77	1.5 ± 0.56	0.39 ± 0.32	0.46 ± 0.33
Normal 2 Closing Cube	0.63 ± 2.17	2.13 ± 0.48	0.85 ± 0.74	0.63 ± 0.47
Normal 2 Peak Systolic Cube	0.63 ± 0.25	2.13 ± 0.25	0.87 ± 0.65	0.75 ± 0.57
Normal 2 Opening Cube	0.63 ± 2.64	2.13 ± 0.51	1.20 ± 0.75	0.70 ± 0.48
Normal 2 Closing Best	0 ± 2.17	1.5 ± 0.48	0.71 ± 0.65	0.57 ± 0.40
Normal 2 Peak Systolic Best	0 ± 0.25	1.5 ± 0.25	0.41 ± 0.34	0.43 ± 0.33
Normal 2 Opening Best	0 ± 2.64	1.5 ± 0.53	0.63 ± 0.42	0.85 ± 0.57
Flail Closing Cube	0.63 ± 3.55	2.13 ± 0.71	0.87 ± 0.66	0.98 ± 0.75
Flail Peak Systolic Cube	0.63 ± 0.25	2.13 ± 0.25	0.62 ± 0.43	0.73 ± 0.49
Flail Opening Cube	0.63 ± 1.73	2.13 ± 0.46	1.08 ± 0.57	0.89 ± 0.63
Flail Closing Best	0 ± 3.55	1.5 ± 0.71	0.46 ± 0.43	0.69 ± 0.72
Flail Peak Systolic Best	0 ± 0.25	1.5 ± 0.25	0.45 ± 0.36	0.32 ± 0.42
Flail Opening Best	0 ± 1.73	1.5 ± 0.46	0.55 ± 0.38	0.59 ± 0.48
Billowing Closing Cube	0.63 ± 1.76	2.13 ± 0.47	1.95 ± 1.21	2.15 ± 2.08
Billowing Peak Systolic Cube	0.63 ± 0.25	2.13 ± 0.25	1.51 ± 1.20	2.90 ± 1.92
Billowing Opening Cube	0.63 ± 3.19	2.13 ± 0.62	1.25 ± 1.19	2.54 ± 1.93
Billowing Closing Best	0 ± 1.76	1.5 ± 0.47	0.88 ± 0.63	0.78 ± 0.63
Billowing Peak Systolic Best	0 ± 0.25	1.5 ± 0.25	0.45 ± 0.30	0.64 ± 0.48
Billowing Closing Best	0 ± 3.19	1.5 ± 0.62	0.79 ± 0.89	0.86 ± 0.89

Table 5.4 Summary of Total and RMS Errors for Dynamic Cases

	Measured Total Error (mm)	RMS Error (mm)
Normal 1 Closing Cube	0.99 ± 0.75	1.24
Normal 1 Peak Systolic Cube	0.93 ± 0.51	1.06
Normal 1 Opening Cube	0.99 ± 0.76	1.25
Normal 1 Closing Best	0.45 ± 0.37	0.58
Normal 1 Peak Systolic Best	0.33 ± 0.25	0.41
Normal 1 Opening Best	0.41 ± 0.32	0.52
Normal 2 Closing Cube	0.77 ± 0.66	1.01
Normal 2 Peak Systolic Cube	0.83 ± 0.62	1.03
Normal 2 Opening Cube	1.05 ± 0.71	1.26
Normal 2 Closing Best	0.65 ± 0.73	0.97
Normal 2 Peak Systolic Best	0.42 ± 0.34	0.54
Normal 2 Opening Best	0.71 ± 0.49	0.86
Flail Closing Cube	0.93 ± 0.71	1.17
Flail Peak Systolic Cube	0.68 ± 0.46	0.82
Flail Opening Cube	1.02 ± 0.59	1.18
Flail Closing Best	0.62 ± 0.65	0.89
Flail Peak Systolic Best	0.38 ± 0.40	0.55
Flail Opening Best	0.56 ± 0.41	0.69
Billowing Closing Cube	2.01 ± 1.59	2.56
Billowing Peak Systolic Cube	1.97 ± 1.61	2.54
Billowing Opening Cube	1.71 ± 1.62	2.35
Billowing Closing Best	0.85 ± 0.63	1.06
Billowing Peak Systolic Best	0.52 ± 0.39	0.65
Billowing Closing Best	0.82 ± 0.89	1.21

5.5 Mesh Refinement Results

Three cases from the dynamic valve results were analyzed using the refinement techniques described in section 5.2. After each refinement method was applied to an under-sampled data set, the marker data and virtual model were compared after best fit. The results from each case follow the same format as those for the dynamic results. The following is an outline of the mesh refinement results and corresponding tables.

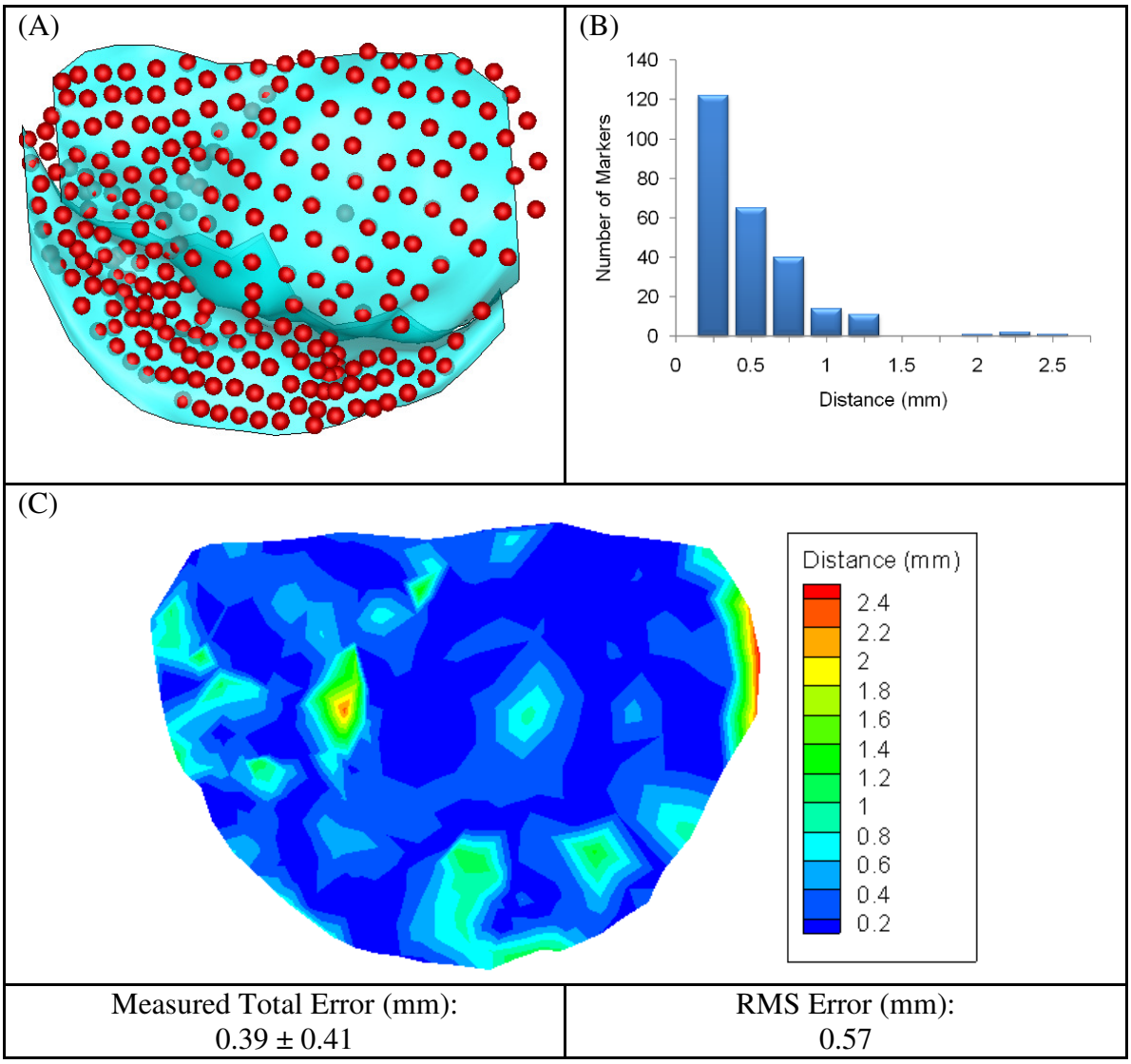
1) Numerical Results from All Mesh Refinement Cases (Table 5.5)
2) Closed Normal Valve 1
    a. Valve after Loop Refinement and Best Fit (Figure 5.31)
    b. Valve after Butterfly Refinement and Best Fit (Figure 5.32)
    c. Valve after Interslice Refinement and Best Fit (Figure 5.33)
3) Closed Valve with Flail Leaflet
    a. Valve after Loop Refinement and Best Fit (Figure 5.34)
    b. Valve after Butterfly Refinement and Best Fit (Figure 5.35)
    c. Valve after Interslice Refinement and Best Fit (Figure 5.36)
4) Closed Valve with Billowing Leaflet
    a. Valve after Loop Refinement and Best Fit (Figure 5.37)
    b. Valve after Butterfly Refinement and Best Fit (Figure 5.38)
    c. Valve after Interslice Refinement and Best Fit (Figure 5.39)

Table 5.5 Numerical results from all mesh refinement cases

		Original	Loop Refinement	Butterfly Refinement	Interslice Refinement
Normal Valve 1 (Best)	Mean Distance (mm)	0.35 ± 0.35	0.39 ± 0.41	0.34 ± 0.34	0.35 ± 0.34
	RMS Distance (mm)	0.50	0.57	0.48	0.48
Valve with Flail Leaflet (Best Fit)	Mean Distance (mm)	0.38 ± 0.40	0.46 ± 0.44	0.45 ± 0.42	0.44 ± 0.43
	RMS Distance (mm)	0.55	0.63	0.62	0.62
Valve with Billowing Leaflet (Best Fit)	Mean Distance (mm)	0.58 ± 0.47	0.63 ± 0.53	0.60 ± 0.59	0.59 ± 0.61
	RMS Distance (mm)	0.74	0.82	0.84	0.85

Table 5.5 contains the results for each of the refinement methods for normal valve 1, the valve with a flail leaflet, and the valve with a billowing leaflet. For each case, the Loop refinement method had the largest mean distance. The Butterfly and Interslice refinement methods had very similar results within 0.01 mm of each other for each condition and measure. The distributions and distance maps for these two methods were also very similar for each condition. The detailed results for each condition and refinement method are presented in the following figures.

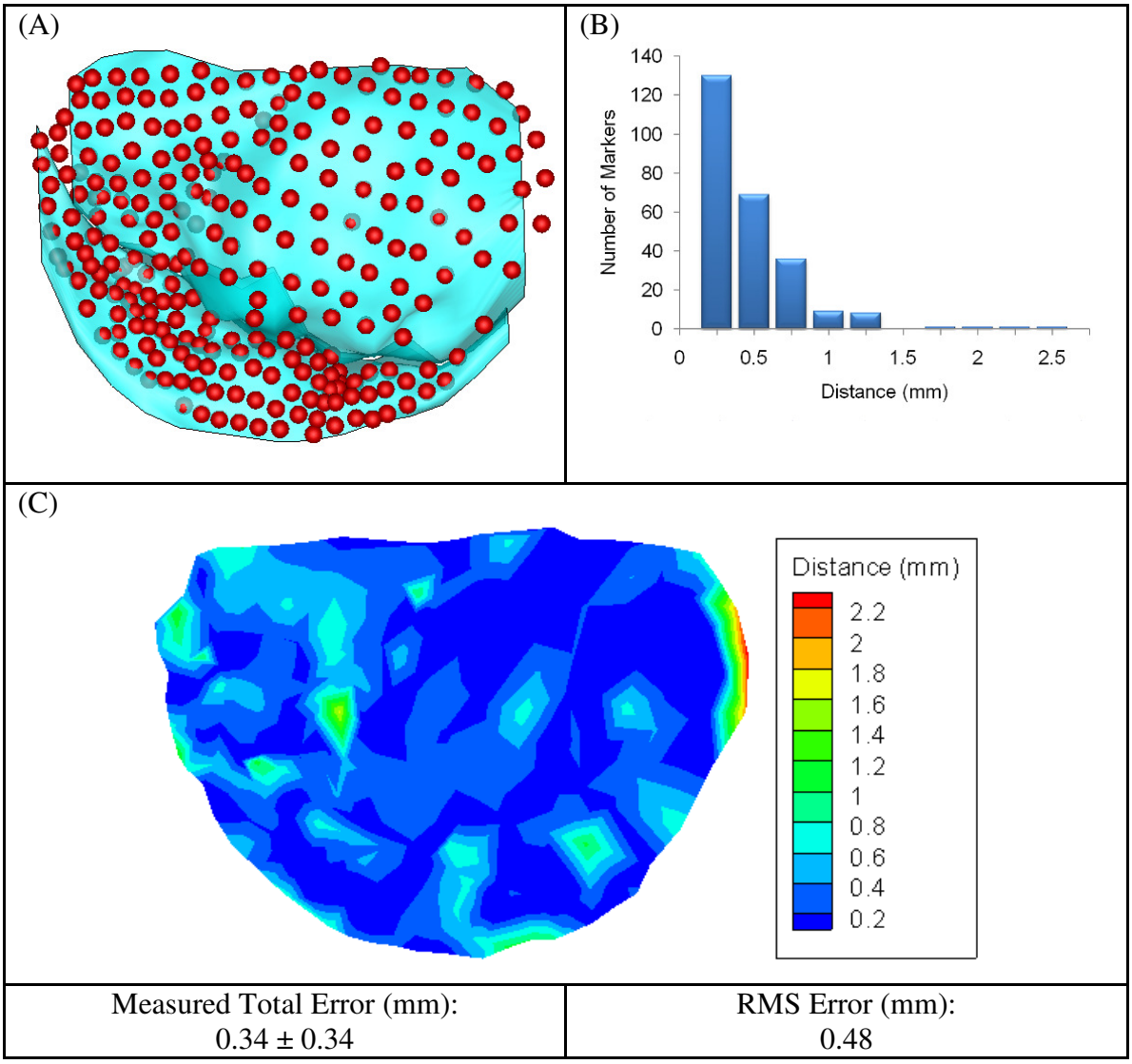
Normal Valve 1 after Loop Refinement



Measured Total Error (mm):	RMS Error (mm):
0.39 ± 0.41	0.57

Figure 5.31 Results for normal valve 1 after Loop refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

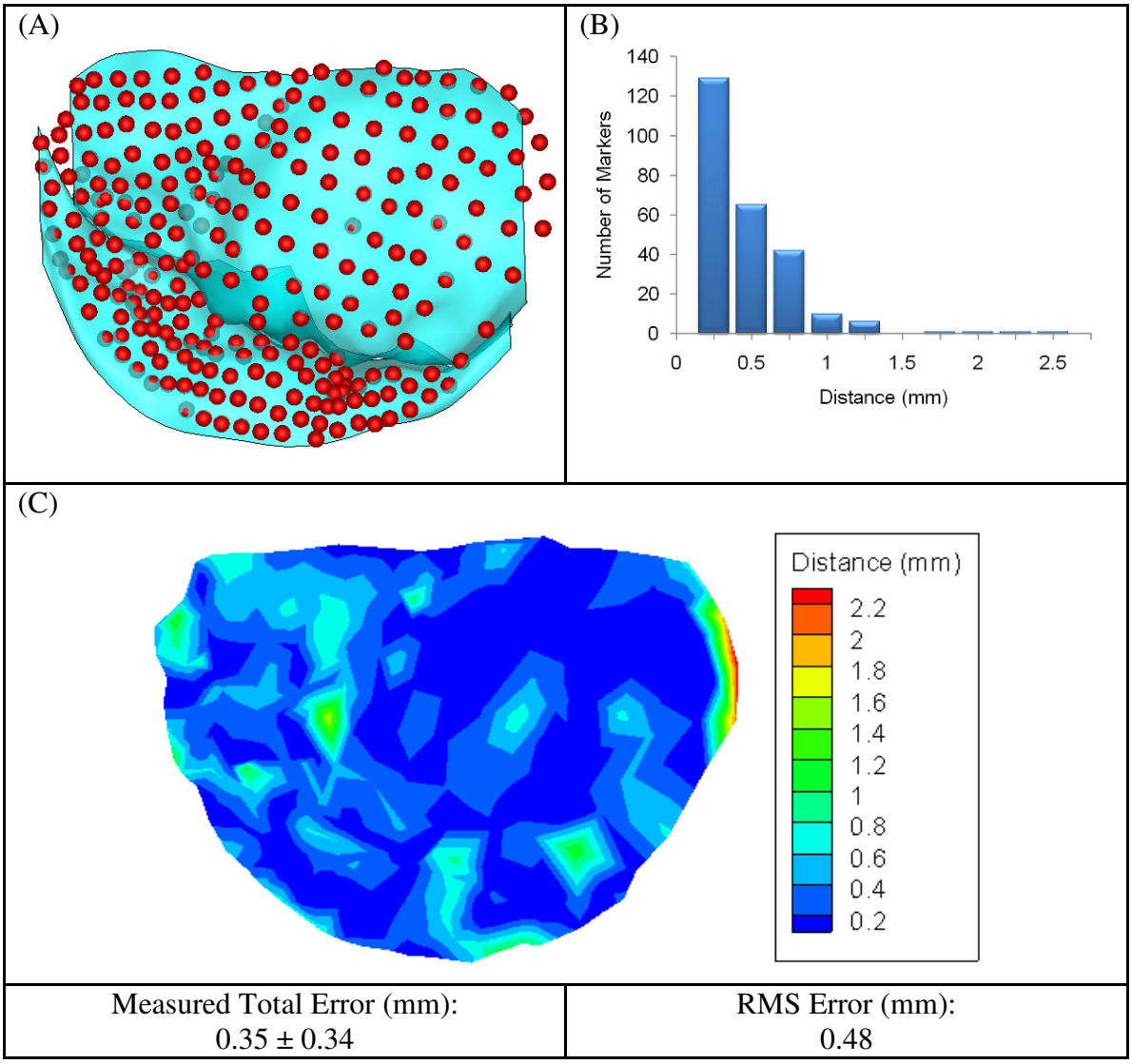
Normal Valve 1 after Butterfly Refinement



Measured Total Error (mm):	RMS Error (mm):
0.34 ± 0.34	0.48

Figure 5.32 Results for normal valve 1 after Butterfly refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Normal Valve 1 after Interslice Refinement



Measured Total Error (mm):	RMS Error (mm):
0.35 ± 0.34	0.48

Figure 5.33 Results for normal valve 1 after Interslice refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Flail Valve after Loop Refinement

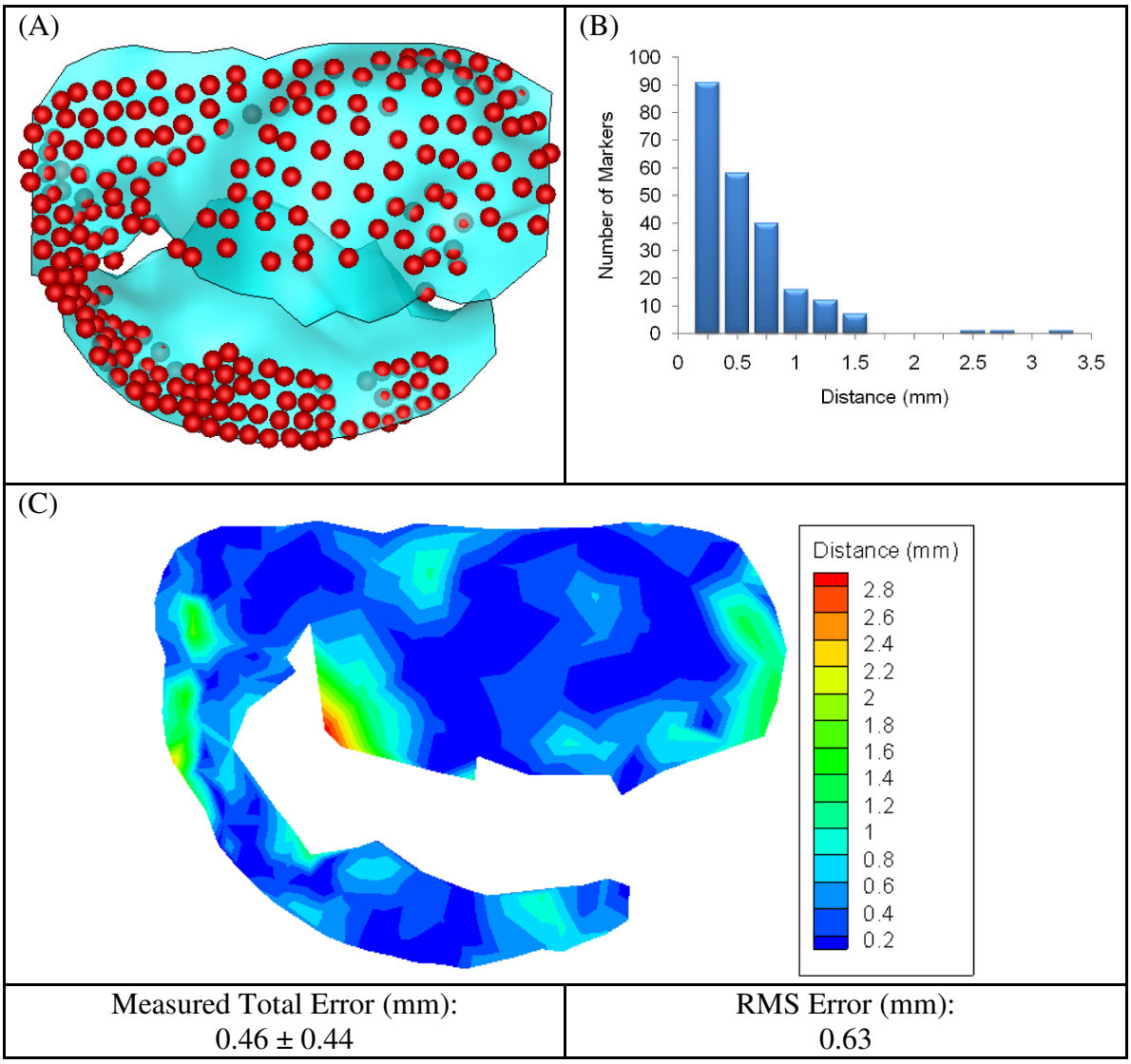
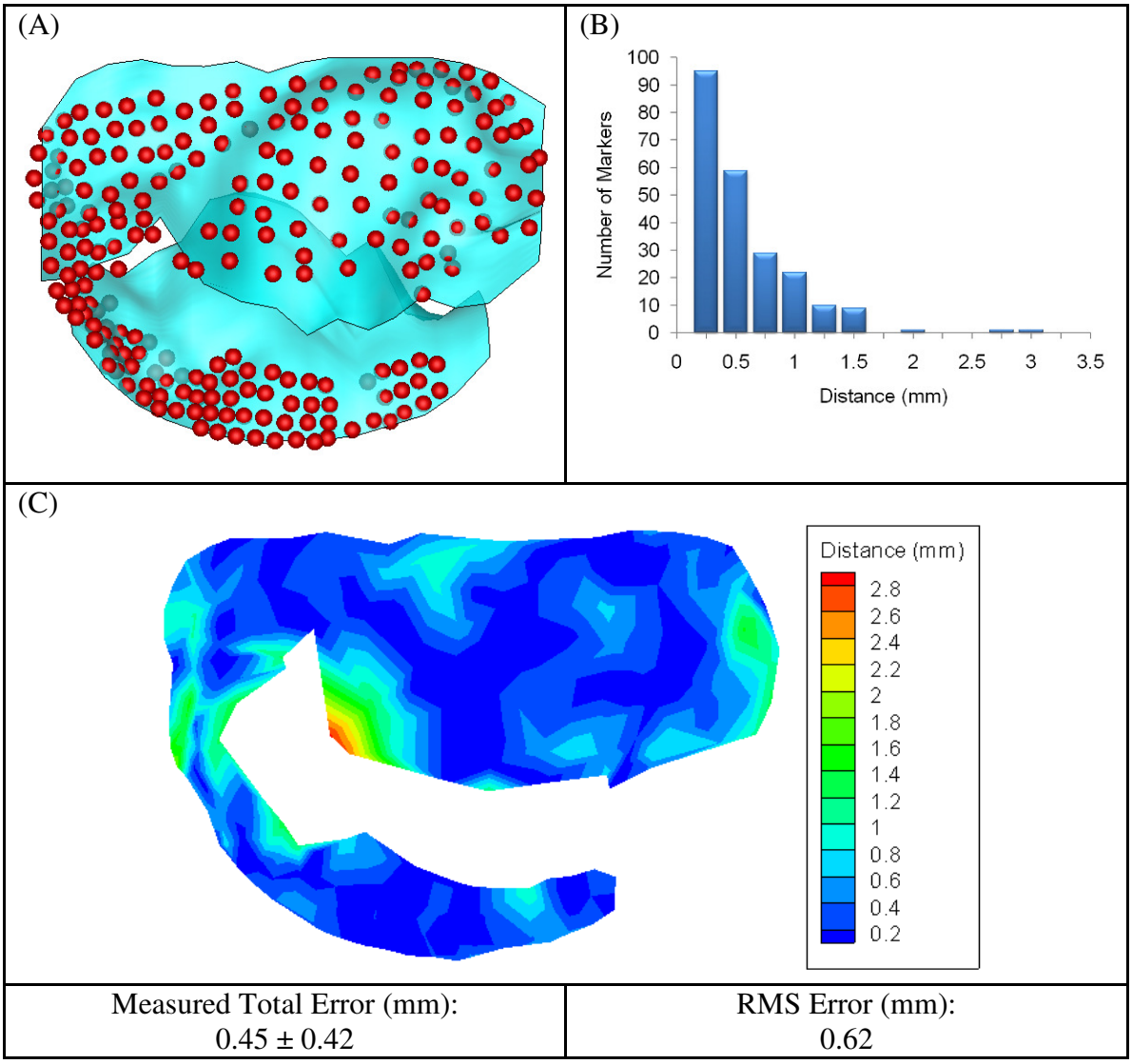


Figure 5.34 Results the valve with a flail leaflet after Loop refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

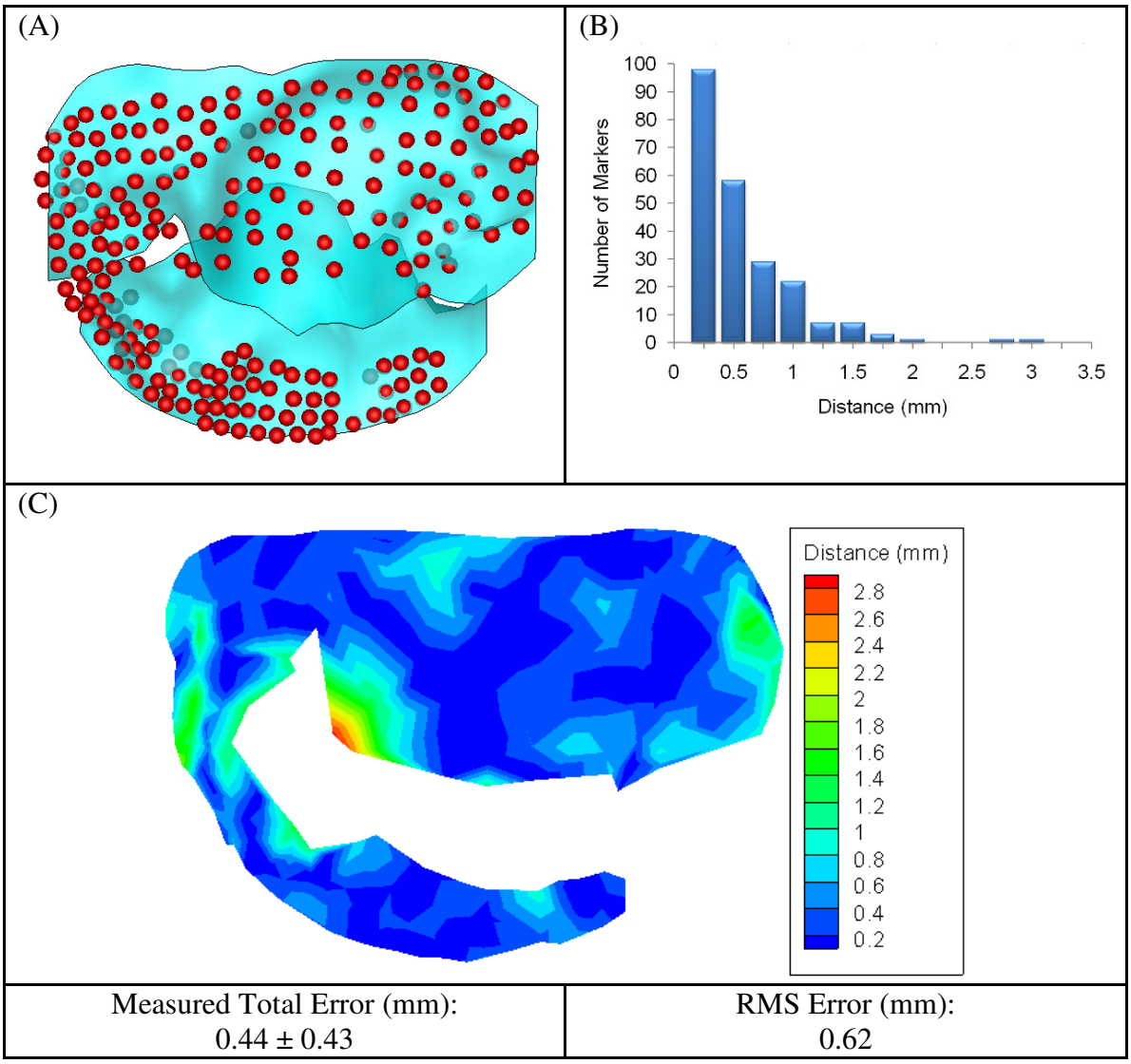
Flail Valve after Butterfly Refinement



Measured Total Error (mm):	RMS Error (mm):
0.45 ± 0.42	0.62

Figure 5.35 Results the valve with a flail leaflet after Butterfly refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

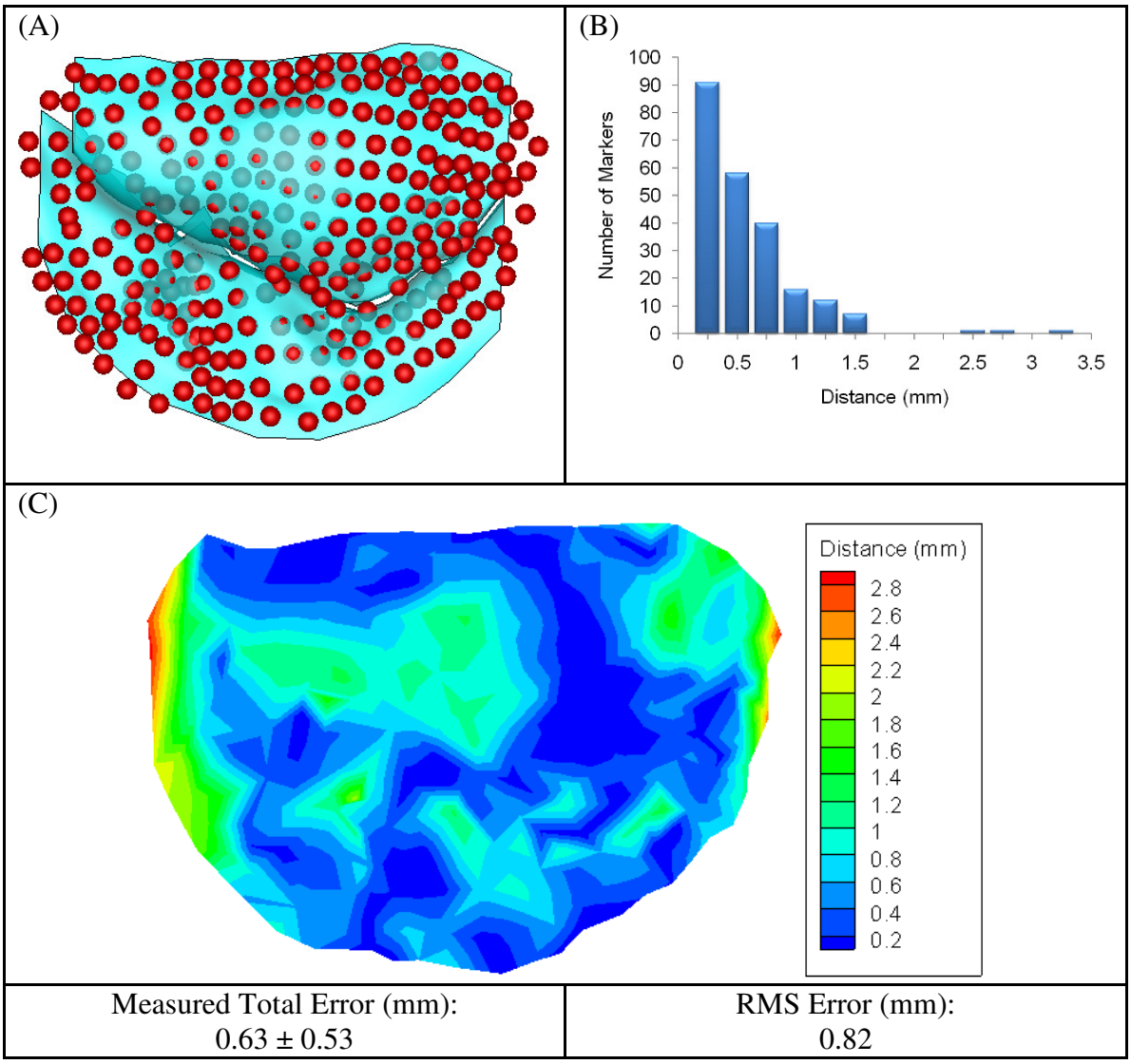
Flail Valve after Interslice Refinement



Measured Total Error (mm):	RMS Error (mm):
0.44 ± 0.43	0.62

Figure 5.36 Results the valve with a flail leaflet after Interslice refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

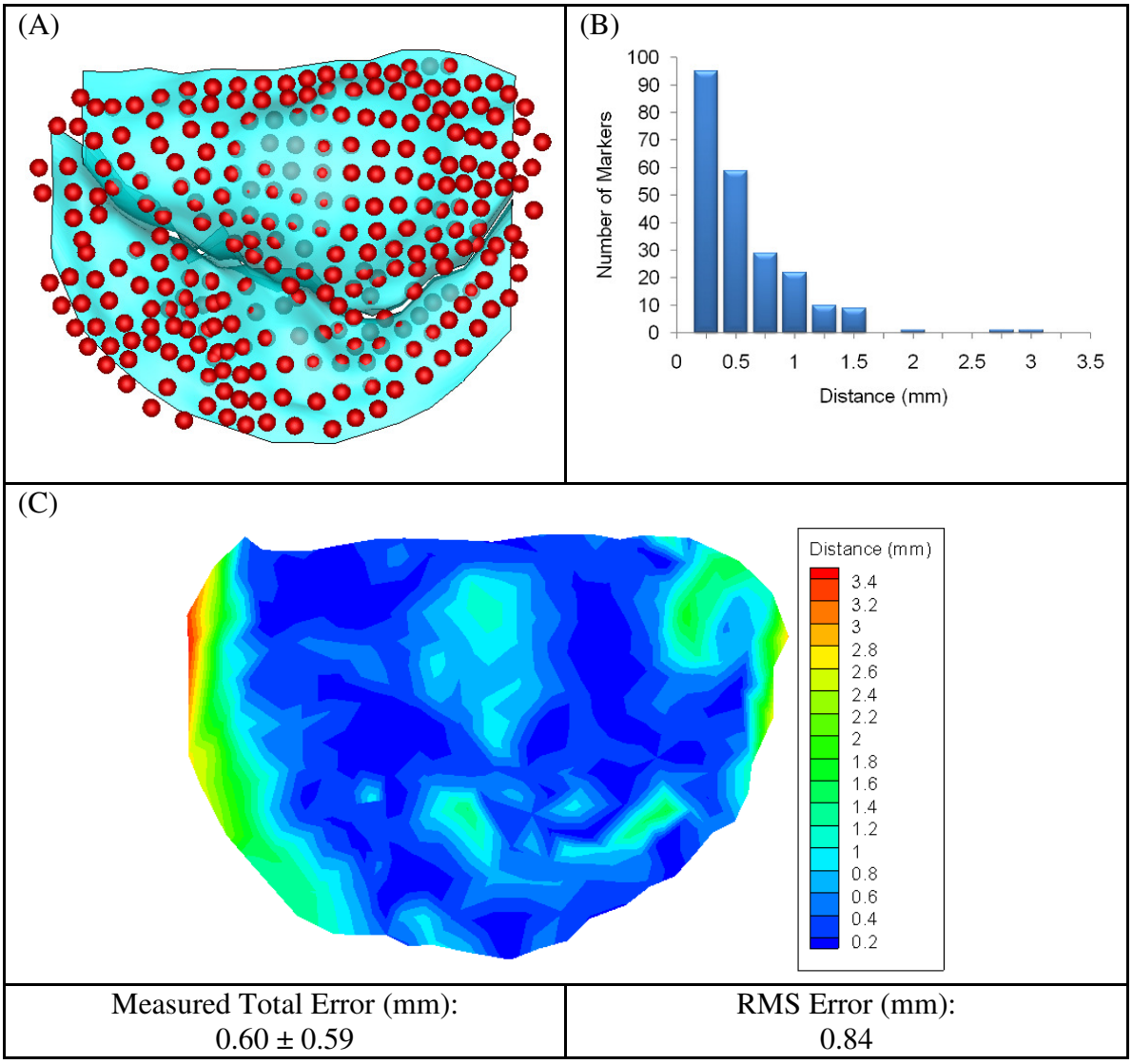
Billowing Valve after Loop Refinement



Measured Total Error (mm):	RMS Error (mm):
0.63 ± 0.53	0.82

Figure 5.37 Results the valve with a billowing leaflet after Loop refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

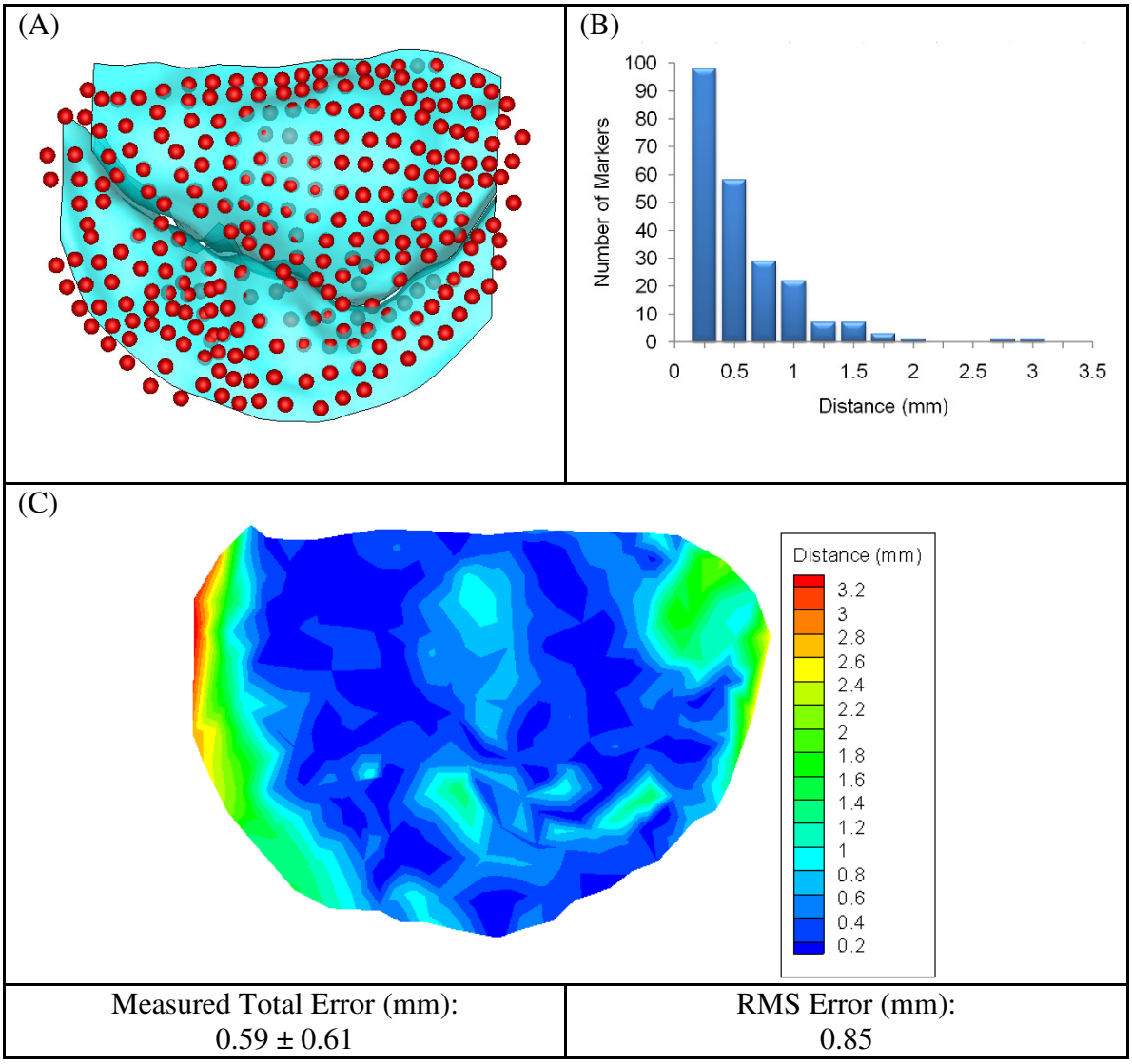
Billowing Valve after Butterfly Refinement



Measured Total Error (mm):	RMS Error (mm):
0.60 ± 0.59	0.84

Figure 5.38 Results the valve with a billowing leaflet after Butterfly refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

Billowing Valve after Interslice Refinement



Measured Total Error (mm):	RMS Error (mm):
0.59 ± 0.61	0.85

Figure 5.39 Results the valve with a billowing leaflet after Interslice refinement and best fit alignment at peak systole. (A): Marker data (red spheres) and virtual model after best fit alignment (blue surface). (B): Distribution of minimum distances from each marker data to virtual model. (C): Distance at each marker point mapped onto a surface created between the marker data points.

CHAPTER 6

DISCUSSION

6.1 Specific Aim 1: Segmentation Method Development

Three different manual segmentation methods were investigated in this study: point-based, line-based, and curve-based. In addition, different triangulation methods were investigated in conjunction with the point and line based segmentation methods. There were also graphical user interface (GUI) improvements made as the segmentation and triangulation methods progressed.

6.1.1 Point Segmentation Method

While the point method was able to generate segmentation data for the leaflets, it was not an optimal solution. This method required the user to estimate the number of points required to adequately capture the leaflet geometry. It also required a user click for every point of the segmentation. The total time required for a complete valve segmentation using this method was approximately 1 hour for each time point. The long segmentation time required for this method was a significant limitation of this method. It would not be practical in a clinical setting to spend an hour for each time point in each patient data set prior to surgery.

In addition, this method also resulted in high variability in the number of segmentation slice. There were no restrictions on the number of points a user could select in each slice. This left it up to the user to decide for themselves how many points were required to represent each leaflet in every slice. A simple shape such as an open leaflet could require only 3 points, while a more complex might require more than 12 points.

The GUI for the point segmentation method also had some limitations. It only displayed a single 2D image at a time. It also only displayed a single time point of the data set at a time. This limited the amount of information the user had about the data set. The user could not examine other planes or time points to confirm their leaflet tracking was correct.

The Tecplot triangulation used in conjunction with the point segmentation method also had some limitations. It produced meshes that wrapped onto themselves as shown in section 4.1.3. This was a result of the z-coordinate of the data being triangulated being ignored by the Tecplot triangulation calculations. Another contributing factor to the poor triangulations was the high variability between neighboring segmentation slices. This resulted in triangles being formed between points that were not in neighboring slices, which did not make physical sense because each segmentation slice is only physically connected to its neighboring slices. Therefore, triangles should only be formed between neighboring sets of segmentation data.

To mitigate the long segmentation time, high point variability between slices, and allow the user to view multiple time points, the line segmentation method was developed. To address the shortcomings of the Tecplot triangulation method, another triangulation method was developed in conjunction with a line segmentation method. GUI improvements were also made in order to provide the user with additional information about the data set that could aid the segmentation.

6.1.2 Line Segmentation Method

The line method used connected points to provide the user with a guide of how many points were required to represent the leaflet geometry. This was designed to ensure that the user would segment enough points to adequately capture the geometry. In addition, the segmentation time was reduced to about 40 to 45 minutes. This reduction was a result of fewer user clicks and less time needed to determine the number of points

necessary for capturing the leaflet geometry. While the reduction in time was an improvement, the line method still required the user to select every segmented point. It also did not eliminate the variability in the number of segmentation points in neighboring slices.

To address the limitations of the Tecplot triangulation method a new triangulation method was developed as part of the line segmentation development. This method iteratively triangulated neighboring slices instead of determining the connectivity of the entire data set at once as done by the Tecplot triangulation method. This eliminated the problem meshes that wrapped onto themselves and triangles being formed between points that were not in neighboring slices. However, this triangulation method did sometimes result in artifacts when the number of points in neighboring slices was unequal as shown in section 4.1.4.

The GUI was also redesigned significantly during the line segmentation development. Instead of a single view of the data set four views were given to the user. This was known as the quad view GUI. A major change from the single view GUI was that the entire DICOM was imported into the segmentation program. This allowed the user to view multiple planes and time points from the data set. The user could now examine the leaflet motion during segmentation, which improved the segmentation quality. In addition, the user could segment multiple time points within the same instance of the program. Along with the three planar views of the DICOM, a 3D plot of the currently segmented points was displayed. These views provided the user with an en face view of the annulus and a commissure-to-commissure view of the leaflets in addition to the segmentation view. The en face annulus view helped guide the segmentation of the annulus by plotting the segmented annulus points on the view in addition to the anterior-posterior segmentation view. However, the commissure-to-commissure view did not provide the user with pertinent information for the segmentation. The 3D plot of the

segmentation data also did not provide the user with additional data to improve the segmentation.

To provide a complete tool for segmenting echocardiography data sets, tools to manipulate the DICOM were developed. These included scaling, rotating, and cropping the data set. Each of these manipulations had a specific objective. Scaling the DICOM to achieve cubic voxels ensured that any rotation operations could be performed without having to take into account variable spacing in each direction of the Cartesian DICOM. This simplified the code for rotation significantly. The rotation code enabled any data set to be oriented in the proper manner for segmentation. This enabled any echocardiography data set to be segmented as long as the entire mitral valve was present in the image. This is important because the purpose of this tool is to segment *in vivo* patient data and valve orientation will vary in patient data. Echocardiography images may also be acquired transapically or transesophageally which would change the orientation of the mitral valve in the image. In addition, there is variability in patient heart size and orientation within the body, which would change the location of the valve in their particular echocardiography image.

The final manipulation made to the DICOM data before segmentation was cropping. Cropping allowed the user to remove parts of the data set that did not include the mitral leaflets in a simple manner. This enabled the user to display on the region of interest during valve segmentation. It also significantly reduced the data set, which in turn reduced memory usage and improved program speed when changing time points or image views.

The line segmentation method made significant improvements over the point segmentation method, but still suffered from some shortcomings. This included a large number of user click required for segmentation, triangulation artifacts from differences in the number of points in neighboring slices, and a GUI that included information that did

provide the user with additional useful information. These shortcomings were address in the curve segmentation method.

6.1.3 Curve Segmentation Method

To reduce segmentation time further and eliminate triangulation artifacts, segmentation using curves was investigated. Three different types of curves were investigated: MATLAB splines, Bezier curves, and J-splines. The use of MATLAB splines proved to be a poor choice for segmentation because the order of the points was only based upon the *x*-coordinate of the points. This method also did not allow for multiple points to have the same *x*-coordinate. Therefore, Bezier curves were investigated because they were dependant on the order in which the points were selected to determine the path of the curve.

Bezier curves were used by Bashein et al[12] for their manual segmentation of a static porcine mitral valve leaflets. Bezier curves provided a method that could generate curves based upon an order set of points which was useful for this application. However, there were substantial drawbacks to using these curves for this application. These were that Bezier curves create a global fit of the control polygon and a different retrofit formula was required to make the curve pass through the selected points for each number of points chosen. The global fit meant that the entire curve changed anywhere the user added a new point to the control polygon. Since the shape of the valve near the annulus is quite different than the shape of the valve near the free edge, a global fit can result in a less efficient segmentation. If the user begins the segmentation at the annulus and moves towards the free edge, using global fit curves will cause the entire segmentation curve to change until the final point is selected. This can mean that the curve created near the annulus at the start of the segmentation is not the same at the end. Therefore, once the entire leaflet was segmented the user would have to go back and adjust the entire curve until the desired result was reached. In addition, Bezier curves do not always pass

through the control points that generate the curves and the formulas to retrofit the curves so they pass through the selected point varies depending on the number of points selected. This would be cumbersome to program and limit the number of points the user could select for their segmentation. Therefore, J-splines were investigated to eliminate these issues.

J-splines proved to be the best method for this application. They were easy to implement in MATLAB and their local fitting characteristics enabled the user to segment the valve near the annulus and have it unaffected by the valve segmentation near the free edge. This also reduced the number of clicks required for the user to segment the valve compared to the line segmentation method. This method resulted in a segmentation time of about 25 minutes per time point, which was far less than the 90 minutes per valve reported by Bashein et al[12]. By significantly reducing the segmentation time, J-splines mitigated one of the two main drawbacks of manual segmentation. The other major drawback to this method was user variability. However, user variability will always be part of a manual method, but in the realm of clinical image processing a manual segmentation performed by an expert is often used as the standard to compare an automatic segmentation against.

J-spline based segmentation may also be useful for other applications such as ventricle, vessel, or tumor segmentation. In these applications J-splines would form closed loops instead of open shapes as in the mitral leaflet segmentation. J-splines can easily create complex shapes that are often present in various types of medical image processing. Its local fitting properties also allow for different sections of complex shapes to be generated without affecting the rest of the curve. In addition, J-splines could be used for the user input in semi-automated methods and as a manual correction for automated methods.

The dual view GUI provided the user with only the views that were required for or improved the segmentation. These were the anterior-posterior view of the leaflets and

the en face view of the annulus. The anterior-posterior view was required for segmenting the annulus and the leaflets and the en face of the annulus provided additional guidance for determining the annulus location. In addition, since only two views were displayed, the size of each image was increased on screen, aiding segmentation further. Therefore, a dual view GUI was the optimal choice for mitral valve segmentation.

6.2 Specific Aim 2: Segmentation Validation

6.2.1 Echo Correction

When acquiring echocardiography images from an *in vitro* environment, refraction and variable speed of sound must be taken into account. This study showed that the errors cannot be corrected with a simple scaling factor because they varied linearly depending on the media properties and exponentially with beam angle and beam radius. The echo correction scheme presented in this study was able to correct the measured distances to within 2.3% or less of their actual distances of 6.320 cm and 5.044 cm for three different atrial chambers. These chambers had varying acrylic thicknesses from 0.297 to 1.232 cm. This showed that the echo correction scheme is robust enough to be applied to chambers of varying geometry while maintaining accuracy, even when the measured error was in excess of 29%.

While the echo correction was good, it was not perfect. The most likely cause of this was the resolution of the 3D echo image which was 0.5 mm. This would create a 0.5 mm cube in which the exact point of the edge selected would be within. Therefore, there could be up to 0.25 mm of error in each direction for the actual point. The assumption of a perfectly flat probe head should have a negligible effect on the correction. The echo probe head had very slight curvature resulting in a slight increase in the ultrasound gel layer thickness at the edges of the probe. Since there was no refraction in the ultrasound gel medium this would have resulted in negligible errors. Still, correcting the data from multiple setups with up to 29% error down to within 2.3% of actual was significant.

The echo correction scheme presented in this study could be applied to any environment in which ultrasound measurements are taken, as long as the geometry and properties of the media involved are known. The scheme can also handle any number of layers because the approach is iterative. The beam path is calculated in the each media from the first entered to the last. This scheme could be particularly useful in calculating flow with 3D ultrasound in an *in vitro* environment. To determine flow using ultrasound, velocities are measured throughout a volume and region of similar velocities is found. The area of this region is then used to calculate the flow rate. Depending on the media and the position of the measured point relative to the ultrasound probe, there could be significant errors in the measured position. Since the errors would be non-linear depending on the distance from the center of the probe, a simple correction scheme would not easily correct these positions. If uncorrected spatial positions are used to calculate the isovelocity region, then the calculated area of this region could have significant errors.

Most *in vitro* flow measurements using 3D ultrasound are taken through a stiff media such as acrylic. Since acrylic has a significantly different speed of sound compared to ultrasound gel and the body, a correction for the position of the measured velocities should be taken into account when calculating isovelocity fields and areas. This will aid in a more accurate flow measurement and correlation to clinical applications of measuring flow in patients using 3D ultrasound.

6.2.2 Normal Valve 1

For normal valve 1 under closing conditions there was good agreement between the estimated and measured errors for both the cube transformation and best fit cases. For the cube transformation the measured errors within the belly and commissure regions were similar at 1.06 ± 0.80 mm and 0.91 ± 0.70 mm respectively. The overall measured error was 0.99 ± 0.75 mm with a range of 0 to 3.36 mm. Figure 5.7B shows the

distribution of the measured distances has a positive skew with 58.3% of the distances measured less than 1 mm and 76.2% less than 1.5 mm. The distance map in Figure 5.7C shows there are concentrations of greater distances in the middle of the posterior leaflet than on the anterior leaflet. This suggests that the cube transformation is not correct and is off by a rotation and possible a translation. This is likely due to the echo data set resolution being unable to capture the corners of the calibration cube precisely.

Both the belly and commissure regions have similar errors at 1.06 ± 0.80 mm and 0.91 ± 0.70, respectively. These results are less than or within one standard deviation of the estimated errors. However, the expected measured commissure error was higher than the belly region at 2.13 ± 0.44 mm and 0.63 ± 1.61 mm respectively. The reason for this discrepancy is most likely that the cube transformation is not perfect and skews the measured values.

When comparing closing normal valve 1 after cube transformation to the best fit case, the best fit has much smaller errors. The measured total error for the best fit case was 0.45 ± 0.37 mm and the RMS error was 0.58 mm compared with a total error of 0.99 ± 0.75 mm and an RMS error of 1.24 mm for the cube transformation case. The measured belly error was 0.42 ± 0.33 mm and the measured commissure error was 0.48 ± 0.42 mm. These are both within less than or within one standard deviation of the corresponding estimated error.

Figure 5.8C does not have concentrations of larger error as seen in Figure 5.7C. For this case, there is a fairly constant error across the valve. This suggests that there are limitations with the cube transformation method, but that the virtual model captures the leaflet geometry well for this case. Figure 5.8B shows that the distribution of measured distances has a strong positive skew and the majority of the distances are less than 1 mm. 90.3% of the markers are within 1 mm of the virtual model and 98.3% are within 1.5 mm. This demonstrates that the virtual model captures the leaflet geometry well for this case.

For normal valve 1 during peak systole after cube transformation qualitative agreement between the marker data and the virtual model can be seen in Figure 5.9A. Figure 5.9B depicts the error distribution between the marker data and the virtual model. The distribution appears to be centered around 1 mm and is approximately symmetric. For this case 55.6% of the markers were within 1 mm of the virtual model and 84.8% were within 1.5 mm. The total error was 0.93 mm and the RMS error was 1.06 mm. The measure belly and commissure errors were similar to the total error at 0.93 ± 0.56 mm and 0.92 ± 0.46 mm, respectively. This suggests that the distances across the valve are similar.

Figure 5.9C illustrates the distribution of the distance between the marker data and the virtual model. Concentrations of higher error occur on the left and bottom of the valve with a gradient toward smaller error on the right side of the valve. This is a strong indication that there is error in the cube transformation. Since there is a gradient of errors, this indicates there is at least an error in the rotation involved in transforming the data set. There could also be some error in the translation of the transformation. The error in the cube transformation is most likely due to the resolution of the echo data set (0.5 mm).

Figure 5.10 shows that the best fit alignment of normal valve 1 at peak systole resulted in all errors decreasing compared to the cube transformation case. Simultaneously, the standard deviations of the measured distances also decreased. The total error was 0.33 ± 0.25 mm and the RMS error was 0.41 mm. The belly and commissure errors were both less than or within one standard deviation of their estimates. The image of the valve (Figure 5.10A) does not clearly show this improvement. However, these improvements can clearly be seen in the distribution of the distances between the marker data and the virtual model (Figure 5.10B) and the corresponding distance map (Figure 5.10C). The distribution has a clearly positive skew compared to the approximate symmetric distribution in the cube case. For the best fit case 98.4% of

the markers are within 1 mm of the virtual model and 100% are within 1.5 mm. The distance map also shows that the distribution of distances across the valve is more even than for the cube transformation case (Figure 5.9C). The regular distribution and the large proportion of markers within 1 mm of the virtual model indicate that the virtual model captures the valve geometry well for normal valve 1 after best fit in the peak systolic case.

The results for normal valve 1 during opening after cube transformation are shown in Figure 5.11. Figure 5.11A shows qualitative agreement between the marker data and the virtual model shapes. The total error for this case was 0.99 ± 0.76 mm and the RMS error was 1.25 mm, which is similar to the closing cube transformation case. The distribution of the distances (Figure 5.11B) has a slight positive skew and 59.7% of the markers were within 1 mm of the virtual model and 73.9% were within 1.5 mm. The distance map (Figure 5.11C) has a similar pattern as the closing case (Figure 5.7C), with greater distance on the posterior leaflet that increase near the annulus. This further supports the notion that the transformation is incorrect because the pattern of distance is similar for both the opening and closing conditions. Figure 5.11C also illustrates an area on the right side of the posterior leaflet where the tissue dye markers could not be seen by both cameras and therefore could not be reconstructed. The measured belly error and commissure error were both less than or within one standard deviation of their estimates.

For normal valve 1 after best fit alignment and during opening, all errors were reduced compared to the cube transformation. All metrics also showed an improvement in the shape match between the markers and the virtual model. The total and RMS errors were reduced to 0.41 ± 0.32 mm and 0.52 mm, respectively. Along with this decrease, a shift in the distance distribution towards zero occurred and a strong positive skew can be seen in Figure 5.12B. For this case 94.1% of the markers are within 1 mm of the virtual model and 99.7% are within 1.5 mm. This indicates that the virtual model matches the shape of the marker data well. The distance map has a much more even distribution of

distances (Figure 5.12C) as seen in the closing and peak systolic best fit cases. The belly and commissure errors are also less than or within one standard deviation of their estimates.

The errors and results from the normal valve 1 for closing, peak systole, and opening after cube transformation all have similar values. The errors and results for normal valve 1 cases after best fit alignment also have similar values. The similarity of the distances across similar cases indicates that the time match for this case is good. In addition, the close match and even distribution of distances for each of the best fit cases demonstrates that the virtual model was able to accurately capture the shape of the marker data in this instance.

6.2.3 Normal Valve 2

For normal valve 2 under closing conditions after cube transformation there is an overall total error of 0.77 ±0.66 mm and an RMS error of 1.01 mm. The measured belly and commissure errors were 0.85 ±0.74 mm and 0.63 ± 0.47 mm, respectively. These were both less than or within one standard deviation of their estimates. The distribution of distances (Figure 5.13B) has a positive skew with 74.2% of markers within 1 mm of the virtual model and 88.6% within 1.5 mm. These are both greater percentages than the comparable normal valve 1 case.

While the total, belly, and commissure errors for this case are all less than the normal valve 1 case, the shape match is not as good as the normal valve 1 case. This can be seen in Figures 5.13A and 5.13C. In Figure 5.13A the marker data points pass behind the anterior and posterior leaflets. This resulted in high distances concentrated in the middle of the valve, which can be seen in the distance map (Figure 5.13C). While the match is fairly close, the match in the middle of the valve is not good with areas of 2 mm or larger distances. This likely means that the time offset in normal valve 2 is greater than that in normal valve 1. This is due to the inability of triggering the echo acquisition

at the same point in the cardiac cycle as the high speed camera acquisition. The cube transformation for normal valve 2 appears to be more accurate than the cube transformation for normal valve 1. This is because the measured total error is less for normal valve 2 cases and the value of the error across the valves is more consistent compared to normal valve 1.

After best fit alignment the errors for normal valve 2 during closing were slightly reduced. The total error was 0.63 ± 0.73 mm and the RMS error 0.97 mm. The average total error decreased by 0.12 mm but the standard deviation also increased by 0.07 mm. The belly and commissure errors both decreased slightly and were less than or within one standard deviation of their predictions. Figure 5.14B shows the distribution of distances associated with this case which has 80.0% of values less than 1 mm and 91.6% less than 1.5 mm. Examining the distance map in Figure 5.14C shows that across all areas of the valve, except the middle, the errors are about 0.4 mm or less. This is an indication that the time match between the marker and echocardiography data for normal valve 2 is worse than normal valve 1. However, given that the overall errors in the cube transformation cases for normal valve 2 are less than the cases for normal valve 2, the cube transformation for normal valve 2 is likely more accurate than normal valve 1.

At peak systole and after cube transformation normal valve 2 had a total error of 0.83 ± 0.62 mm, an RMS error of 1.03 mm, a belly region error of 0.87 ± 0.65 mm, and a commissure region error of 0.75 ± 0.57 mm. Both the belly and commissure region errors were less than or within one standard deviation of their estimates. Figure 5.15A shows that some of the marker data points pass behind the virtual model. The marker distribution (Figure 5.15B) has a positive skew with all values less than 2.5 mm. 68.8% are less than 1 mm and 82.3% are less than 1.5 mm. The distance map in Figure 5.15C indicates that the cube transformation likely has errors in at least the rotation part of the transformation. This is due to the gradient in the errors across the valve, with higher errors present in the lower part of the map and lower errors present in the upper part of

the map. There could also be some error in the cube transformation, but that is unclear from the error map.

For the best fit alignment of normal valve 2 at peak systole the errors were greatly improved with total and RMS errors being 0.42 ± 0.34 mm and 0.54 mm, respectively. The distance map in Figure 5.16C shows that the distribution of errors across the valve is more even than in the cube transformation case. These values, along with the clear shift to smaller errors observed in the distance distribution (Figure 5.16B) suggest that the transformation was slightly off. 92.7% of the markers were within 1 mm of the virtual model and 98.6% were within 1.5 mm. All values were within 1.75 mm. This is a strong indication that the shapes of the marker data and the virtual model are similar.

Normal valve 2 after cube transformation during opening had a total error of 1.05 ± 0.71 mm and an RMS error of 1.26 mm. The belly region had higher error than the commissure region, which is consistent with the closing case. This further supports the notion that there is a time mismatch for this data set. Figure 5.17A shows that there are many marker data points that are behind the valve, suggesting a time mismatch between the compared data sets. The time mismatch is a result of the echo acquisition being unable to sync exactly with the high speed camera acquisition. However, the measured errors are still less than or within one standard deviation of their estimates. In addition, the best fit alignment case improved all error measures, as it did in the closing case.

Overall, normal valve 2 demonstrated that it had a closer match than normal valve 1 for its cube transformation, but it had a higher error for the time mismatch. While this error was higher than for normal valve 1 it is still within the estimated error. For the peak systolic case, there was very good shape match between the marker data and the virtual model. Given this, the program was able to capture the geometry well in this instance.

6.2.4 Valve with Flail Leaflet

The valve with a flail leaflet was compared during closing, peak systole, and closing after cube transformation and best fit alignment. For closing after cube transformation the total error was 0.93 ± 0.71 mm. This is in line with the cube transformation measurements from normal valve 1. The distribution of distances (Figure 5.19B) has a positive skew with 62.0% of values being less than 1 mm and 81.9% less than 1.5 mm. Comparing the distance map (Figure 5.19C) and the overlay (Figure 5.19A), there appears to be a slight rotation by which the transformation is off. This is because the left side of the valve has a cluster of points below the virtual model and the right side has a cluster above it. In the middle of the anterior leaflet is a small group of points with high error. These points are most likely due to the cycle-to-cycle variability of the flail leaflet model. Since the leaflets were completely unrestricted by the chordae, the leaflets did not achieve a similar position at the same time points in each cycle.

The best fit for this case resulted in all errors being reduced compared to the cube transformation and were less than or within one standard deviation of their estimated values. With 80.8% of markers with 1 mm of the virtual model and 91.0% within 1.5 mm, there was good agreement between the two data sets. Figure 5.20C depicts the distance map. This distance map indicates a consistent error across the valve except for a few locations. Given this information the cube transformation was likely off and the cycle-to-cycle variability of the flail leaflet model contributed to groups of larger errors on the valve.

The peak systolic case after cube transformation had smaller errors than the closing case. This suggests there was some time mismatch between the marker and echo data sets that were compared. The total error of this case was 0.68 ± 0.46 mm. From the overlay (Figure 5.21A) there is good qualitative agreement between the markers and the virtual model. The distance map in Figure 5.21C illustrates that there is a small area of large error on the anterior (flail) leaflet and then larger groups of error less than 0.5 mm

on the anterior leaflet and errors around 1 mm on the posterior leaflet. This suggests that there is a gradient across the valve and that the cube transformation is not perfect.

The peak systolic best fit alignment case had good shape agreement. The total error is 0.38 ± 0.40 mm and the RMS error is 0.55 mm. The distance map in Figure 5.22C illustrates the uniformity of the distances across the valve. This clearly demonstrates that the virtual model captured the geometry of the leaflets. The small area of higher error near the left of the anterior (flail) leaflet is likely due to the cycle-to-cycle variability within the flail leaflet model. The uniformity also supports the idea that the cube transformation was off by some rotation and translation.

The cube transformation and best fit alignment for the valve with flail leaflet during opening provide similar insight as the closing case. Comparing the distance maps between the two (Figures 5.23C and 5.24C) show that the cube transformation is not exact and there is a slight time mismatch between the marker data and the virtual model.

Overall, the valve with a flail leaflet was able to capture the geometry in each case. The average errors were all less than or within one standard deviation of the estimates. The best fit alignment cases demonstrated that the shapes of the marker data and the virtual models matched well.

6.2.5 Valve with Billowing Leaflet

The billowing leaflet case provided similar insight as the other three cases. For the closing valve after cube transformation the distance map (Figure 5.25C) shows a clear pattern that the transformation is off. A gradient of lower error from the posterior leaflet to higher error in the top right of the anterior leaflet is clearly visible. For this case the errors were higher than any previous cube transformation case. The total error was 2.01 ±1.59 mm and the RMS error was 2.56 mm. Only 31.8% of the markers were within 1 mm of the virtual model and only 48.9% were within 1.5 mm.

After the best fit alignment, the total error dropped to 0.85 ± 0.63 mm and the RMS error to 1.06 mm. 65.7% of the errors were less than 1 mm and 83.0% were less than 1.5 mm. Figure 5.26C depicts the distance map for this case. Most of the valve has an error around 0.5 mm, with a few areas near the middle that are slightly higher. This indicates that the transformation was not exact and that there is probably a degree of time offset between the compared data sets.

The peak systolic cube transformation and best fit cases draw similar conclusions. The total error for the cube transformation is high at 1.97 ± 1.61 mm with a clear gradient of errors from the posterior leaflet to the top right of the anterior leaflet (Figure 5.27C). After best fit alignment, the errors drop dramatically to 0.52 ± 0.39 mm. In addition, the error map in Figure 5.28C has a nearly completely uniform error across the entire valve. This indicates that the shape of the marker data and the virtual model for this case are extremely close. The percentage of markers within 1 mm of the virtual model was 88.2% for this case and 97.6% were within 1.5 mm.

The opening case for the valve with a billowing leaflet supports the ideas from the closing case that the cube transformation is off and that there is also some time mismatch between the marker data and the virtual model. These errors combine to yield a total error of 1.71 ± 1.62 mm for the cube transformation case. In the best fit alignment case the total error is reduced to 0.82 ± 0.89 mm. This supports the idea that the majority of the error in the cube transformation case is from cube transformation errors and not from time mismatch error.

Since the cube transformation was off much more in the billowing case than any other, there could have been a problem with this acquisition. The calibration cube may have been bumped or the setup slightly moved in between high speed camera and echocardiography acquisitions. This would have changed the orientation of the calibration cube and could account for the large difference in the errors in the billowing cube transformation cases compared to the normal valve and flail leaflet cube

transformation cases. After best fit alignment, the virtual model clearly captured the geometry of each leaflet case within the estimated errors and the peak systolic case had a uniform error across the entire valve. This suggests a strong shape match between the marker data and the virtual model, which indicates that the leaflet geometry was captured well, but there was error in the cube transformation.

6.2.6 Overall Dynamic Valves

For cube transformation cases, over 30% of marker were within 1 mm of the corresponding virtual model and over 45% were within 1.5 mm. The expected offset from the valve thickness was 0.25 to 1.5 mm for all cases. For the best fit alignment cases, over 64% of marker were within 1 mm of the corresponding virtual model and over 80% were within 1.5 mm. For the peak systolic cases over 84% of markers were within 1 mm of their corresponding virtual model and over 95% were within 1.5mm. This demonstrates the effect of the time offset between the marker and echocardiography images in this validation. It also shows that the best fit alignment was better than the cube transformation for comparing the shapes of the marker data and their virtual models. The cube transformation comparisons are worse because of the resolution of the echocardiography data sets.

The calibration cube is 10 mm on each side and the resolution of the echo data is 0.5 mm. The segmentation program assumes that the corner of the calibration cube is exactly in the center of that voxel because there is only a single value given for the entire voxel. Since the precise location of the corner could be anywhere within the voxel, there is an error associated with measuring the calibration cubes from the echo data that is quantifiable. The maximum distance the actual corner location could be from the center of the voxel is in one of the corners, or 0.25 mm in each direction. This yields a maximum error of 0.433 mm in distance for each corner measurement. Given that each primary direction (x, y, z) is given by 2 corner measurements, there could be a maximum

error of 0.866 mm for each primary direction measured on the calibration cube by the echocardiography image. This is up to 8.66% error in each primary direction for the 10 mm cube. This resulted in the calibration cube method producing non-orthonormal bases because of the echocardiography resolution limitations. These non-orthonormal bases could have slightly stretched or skewed the transformed data. Therefore, the best fit alignment method was considered a better comparison of shape similarity between the marker data and the virtual models.

In addition, many of cases did not have the valve completely segmented near the commissures. This is a limitation of the experimental setup because the leaflets were hard to distinguish near the acrylic annulus plate. This should not be an issue when segmenting high quality human data sets.

All average errors were less than or within one standard deviation of their estimated. Overall, the mean distances for the peak systolic best fit cases were near or better than those from Bashein et al[12]. The majority of the errors for the peak systolic best fit cases were less than 1.5 mm. This should provide a surgeon with the necessary accuracy to fully understand the pathologic shape of the leaflets. Therefore, the segmentation method presented in this thesis is on par or better than other manual segmentations methods of closed mitral valves even with the sources of error discussed in the previous paragraphs.

Given a validated method to create dynamic models of the mitral leaflets, there are multiple benefits to diagnosis and surgical planning. Surgeons could examine the dynamics of the valve leaflets as a whole, including how the leaflet area changed throughout the cardiac cycle, the areas of coaptation across the valve, and other characteristics that may aid in surgical planning. In addition, if software to manipulate the virtual models as a surgeon would (cutting, suturing, etc.) was developed, then surgical could interactively test and simulate their repairs before a patient was in surgery.

This could aid surgeons in how to get the best results from a repair and possibly improve patient outcomes.

6.3 Specific Aim 3: Mesh Refinement

Three methods of mesh refinement were examined with the aim of reducing the required segmentation time: Loop, Butterfly, and Interslice. Of the three techniques, the Loop method resulted in the highest mean and RMS distances between the markers and the virtual model. This was most likely because the new points were generated as a combination of only the four nearest points, while the Butterfly and Interslice methods used a larger range of points in interpolating new points. When the mean and RMS distances of the Butterfly and Interslice methods were compared, neither method was clearly better than the other. This was also true when the distributions and distance maps for each method were compared. Since these two methods cannot be distinguished based upon their results alone, other metrics were examined to differentiate the two. A key difference between the methods was that the Interslice method only worked with the segmentation tool, while the Butterfly method can be applied to any triangle mesh. The Interslice method also required that there be the same number of points in each segmentation slice, while the Butterfly method did not. Another advantage is that the Butterfly method generates a new mesh more quickly than the Interslice method. These advantages indicate that the Butterfly method should be used when interpolating the mitral valve leaflet meshes.

For the Butterfly refinement method, all of the mean distances were within 0.07 mm of the corresponding fully sampled virtual model. This shows that when the valve was sampled every 3 mm, there was a small drop in the accuracy of the model. Therefore, Butterfly mesh refinement may be used to segment the mitral valve every 3mm and interpolate the leaflet geometry accurately. This could potentially reduce the segmentation time by half, which makes the segmentation time for a single time point of

the mitral valve around 12.5 minutes, using the J-spline method with Butterfly mesh refinement.

Since Butterfly mesh refinement was shown to interpolate the valve geometry well in order to reduce segmentation time, it may be possible to use this method in other types of medical image segmentation to decrease segmentation time. Vessels, ventricles, and tumors are possible candidates. However, how well Butterfly mesh refinement interpolates these types of structures has yet to be evaluated.

6.4 Clinical Application

By accessing the ability of the segmentation method to capture the leaflet geometry in the simulations of normal and two different disease states, it has been shown that the program is capable of modeling the highly variable geometry present in patient data. Moving forward, this software could be used to model patient data and put that data into a surgical simulator. The virtual models generated by the segmentation program were capable of providing additional measures of leaflet geometry beyond the normal evaluation, including leaflet area and coaptation area. The shape of the annulus and each leaflet is also provided by the virtual models. In addition, the distribution of coaptation length across the valve could be measured, which could aid in surgical planning. A surgeon could focus their repair upon increasing coaptation length in the areas that it is smallest. It could also enable a more complete evaluation of repair efficacy after surgery by providing a complete picture of the coaptation length across the valve instead of in a single plane. Another measurement that could be possible from the virtual models is leaflet curvature using the method presented by Ryan et al[14]. This may provide insight as to whether or not the repair restored normal leaflet shape, which may affect long-term durability of the repair. However, a detailed study of patient data would be needed to examine that effect.

Characterizing the mitral annulus and leaflet geometry may also assist in determining the underlying cause of valve failure. It has been shown that the shape of the leaflets and annulus affect the leaflet stresses[51]. In addition, accurate virtual models could provide boundary conditions for finite element analysis and fluid structure interaction simulations[52, 53]. In the future, it may also be possible to use these virtual models to assess new surgical techniques to treat mitral valve dysfunction[54, 55].

By accessing the ability of the segmentation method to capture the leaflet geometry of simulations of normal and two different disease states, it has been shown that the program is capable of modeling the highly variable geometry present in patient data. Moving forward, this software could be used to model patient data and put that data into a surgical simulator.

6.5 Limitations

Since this study was performed in an *in vitro* environment, there are some unavoidable limitations. While the *in vitro* setup has some limitations in simulating the exact conditions of the heart[3], it also did not allow for the valves to be segmented close to the commissures of the valves. However, this would not be an issue *in vivo* with high quality patient images. For the marker data specifically there were two limitations. The entire cardiac cycle could also not be compared, because there the marker data is not available for the whole cycle. Also, the dual camera stereo photogrammetry technique had increased error for markers near the commissures because they were farthest from the calibration cube.

For the comparison between the echocardiography and high speed camera data sets there were two limitations. First, the echo and marker data could not be triggered at the same instance in the cardiac cycle. This lead to a time mismatch between compared frames. The second was that the echocardiography and high speed camera images could not be acquired during the same cardiac cycle. This was because the ultrasound probe

had to be placed directly into the line of sight of the high speed cameras, blocking their view of the valve.

Finally, there were also two limitations associated with the segmentation itself. Firstly, it was a manual approach that can be affected by user variability, but a user with expertise in mitral valve echocardiography should be able to segment the valve accurately. Secondly, the echo segmentation and corrections used echocardiography data after the original DICOM had been converted into a Cartesian DICOM. For the best results, the correction should be applied to the raw echo image. This is because there could be small errors caused by rounding when converting between coordinate systems.

CHAPTER 7

CONCLUSIONS

This study developed and validated a tool capable of capturing and displaying mitral valve leaflet geometry under normal and diseased conditions. Real-time 3D echocardiography (RT3DE) images of *in vitro* mitral valves were manually segmented using J_0 splines and virtual leaflet models were constructed. The use of these splines instead of user selected points or lines reduced segmentation time. Stereo photogrammetry was used to determine the shapes of the mitral leaflets under normal hemodynamic conditions and was utilized to validate the virtual models.

It was found that the virtual models were able to capture the valve geometry accurately for a variety of test conditions. For the best fit peak systolic cases, over 95% of the markers were within 1.5mm of the virtual models, which indicates that the virtual models captured the valve geometry well. Over 80% of markers were within 1.5mm of the virtual models for dynamic cases, and the comparison was still within the possible error from matching the peak systolic frames.

Of the mesh refinement methods compared, the Butterfly refinement was the best at interpolating mitral valve leaflet geometry. While the Butterfly and Interslice methods were very similar in how well they interpolated the leaflet geometry, the Butterfly method was faster and could be applied to any mesh. Therefore, the Butterfly mesh refinement method could be used to reduce the number of segmentation slices required to capture leaflet geometry accurately.

This study also developed and validated a method to correct *in vitro* ultrasound measurements for distortions created by the acrylic chamber the ultrasound measurements were taken through. The method calculated the actual location of an ultrasound measurement by calculating the actual beam path taking refraction and speed

of sound differences into account. This method was shown to be robust by correcting the same measurements from chambers with varying dimensions. The *in vitro* correction method presented in this thesis could correct any *in vitro* ultrasound measurements in which speed of sound and refraction errors would occur.

CHAPTER 8

RECOMMENDATIONS & FUTURE WORK

The tool presented in this study is a first step towards developing a platform for mitral valve surgical planning. There are steps that should be taken to improve the segmentation tool before it can be used to develop the surgical planning platform. A semi-automated or automated segmentation method should be developed to build upon the manual method developed in this study. This should decrease segmentation time and reduce the amount of user variability in the segmentation process. Another improvement to the segmentation tool that should be pursued is the integration of papillary muscle position and a prediction of chordal insertion that could be used with *in vivo* data. A finite element analysis study could be developed with the created virtual models and validated against experimentally measured parameters such as strain in different regions of each leaflet. A mass-spring model could also be used to predict dynamic motion of the valve and validated against actual models generated at different time points.

To further develop the surgical planning platform, an environment to perform virtual surgery the virtual models should be developed. Methods to perform surgical interventions on the virtual models should be developed including cutting and suturing. Finally, a method to simulate valve closure after surgical manipulation should be developed and integrated into the surgical planning environment.

The J_0 spline based manual segmentation presented in this study could also be applied to other types of segmentation, including left and right ventricles and blood vessels. The only modification required would be to use closed curves instead of open curves. This would eliminate the need for adding additional endpoints as described in section 4.1.5.3. Furthermore, this method could also be used to develop a semi-automated segmentation method for valves, ventricles, and vessels. In this case, the

manual curve segmentation could be used to provide an initial input to the automated portion of the segmentation. This would reduce the time required for the manual portion of the segmentation compared to point and line based manual methods.

Studies to improve the mesh refinement method should also be done. These would include comparing other methods of mesh refinement to attempt to improve upon the Butterfly method. Also, to determine the optimal spacing for a given mesh refinement method more details studies on the effect of segmentation density compare to accuracy would be required.

In addition to segmentation development, there are a number of studies that should be pursued with the current tool. These will be divided into *in vitro* and *in vivo* studies.

For *in vitro* studies:

1) Compare virtual model curvature to strain data acquired using dual camera stereophotogrammetry for normal and diseased valves.
2) The virtual models could be used to study the effect of annular dilation and papillary muscle displacement on the leaflet kinematics, including the time required for leaflet closure and leaflet shape throughout the cardiac cycle.
3) The virtual models could be used to study the effect of annular dilation and papillary muscle displacement on coaptation area and the distribution of coaptation length across the valve. The distribution of coaptation length should be compared to regurgitation jet location. In addition, the tenting height, area, and volume should be compared. This may provide further insight to clinicians for choosing one surgical intervention over another.

For *in vivo* studies:

1) The virtual models could be used to study leaflet kinematics and coaptation area of normal and diseased valves *in vivo*.
2) The virtual models could be used to study changes in leaflet kinematics and coaptation area at varying stages in human or animal development. This could aid in understanding the role of valve dynamics in development of the mitral valve apparatus.

APPENDIX A

SEGMENTATION TOOL USER MANUAL

The procedure for acquiring and processing an echocardiography data set for segmentation is as follows:

1) Acquire 3D echocardiography image using a Philips iE33 echo machine.
    a. Ensure the mitral valve can be seen in the image when acquired at a 50% overall gain setting. This is because the Cartesian DICOM is always exported at 50% gain.
    b. Live 3D or Full Volume images are acceptable.
2) Transfer the echo images from the iE33 machine to a computer that has QLAB with Cartesian Export enabled.
3) Open the 3D echo to be segmented in QLAB.
4) Click the export image file button. 
5) In the *Save As* dialog select file type to be Cartesian DICOM (3DDCM) (Figure A.1).

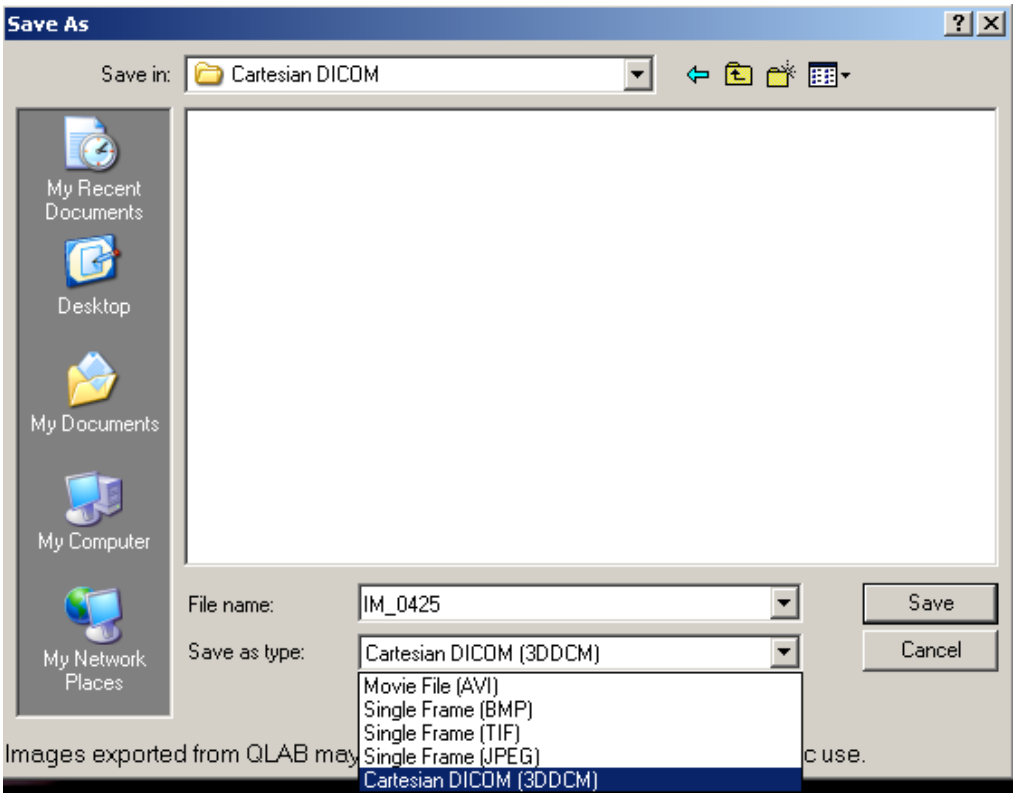


Figure A.1 Save as dialog for saving the Cartesian DICOM.

6) Move the Cartesian DICOM to the rotateMV folder.

7) Read the Cartesian DICOM into MATLAB using *readDicom3Dedited* (Figure A.2).

   [Ex: >> im_0425 = readDicom3Dedited('IM_0425.dcm')]

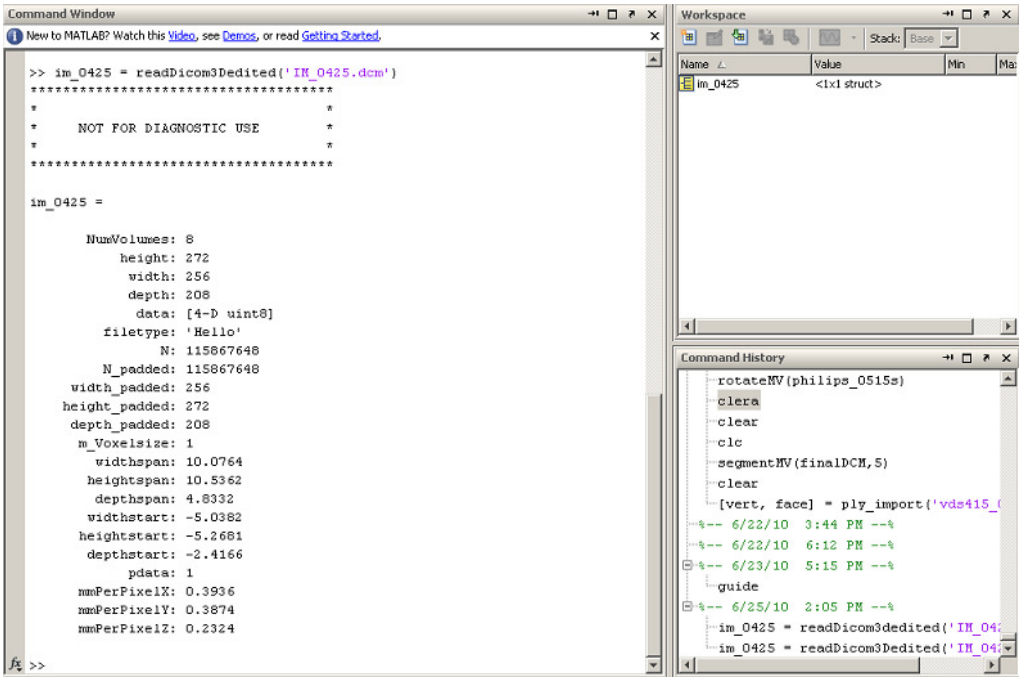


Figure A.2 Example dialog when importing Cartesian DICOM into MATLAB.

8) Save the Cartesian DICOM variable using the save dialog.

   [Ex: save im_0425 im_0425]

9) Scale the Cartesian DICOM to 0.5 mm voxels using *scaleDicom*.

   [Ex: im_0425s = scaleDicom(im_0425);].

   The function will print its progress as an array of 3 numbers. The first number is the section of code it is in, there are two total. The second number is the slice being scaled. The third is the point in time being scaled. If a different size voxel is desired, the mmScale variable within the *scaleDicom* function can be changed.

10) Save the scaled Cartesian DICOM variable using the save dialog.

    [Ex: save im_0425s im_0425s].

11) Import the scaled Cartesian DICOM variable into the *rotateMV* function.

    [Ex: rotateMV(im_0425s)].

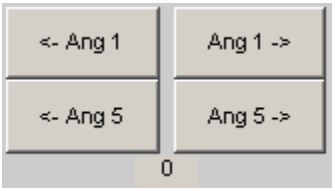
12) The *rotateMV* GUI will display 3 planes of the Cartesian DICOM at an instant in time and provide buttons to orient the data set for segmentation. The plane shown

can be changed using the change slice buttons. All operations affect the top-left view in the *rotateMV* program.

13) Functionality to manipulate the Cartesian DICOM within the *rotateMV* program includes:

a. Rotate by 90˚ with respect to the top-left view clicking 

.

b. The data set can be rotated by an angle by first clicking 

to adjust the angle in the top-left image and then once the angle desired is reached, clicking 

to rotate the entire data set by that angle.

c. The views can be switched by clicking 

.

d. Clicking 

will allow the user to select two points to create a rectangle for cropping on the top-left image. Clicking 

again will allow the user to adjust the rectangle. Once the final rectangle for cropping is selected, clicking 

will crop the data set.

14) The top-left image should contain an anterior-posterior view of the mitral leaflets and the top-right image should contain the annulus plane as shown in the following image (Figure A.3).

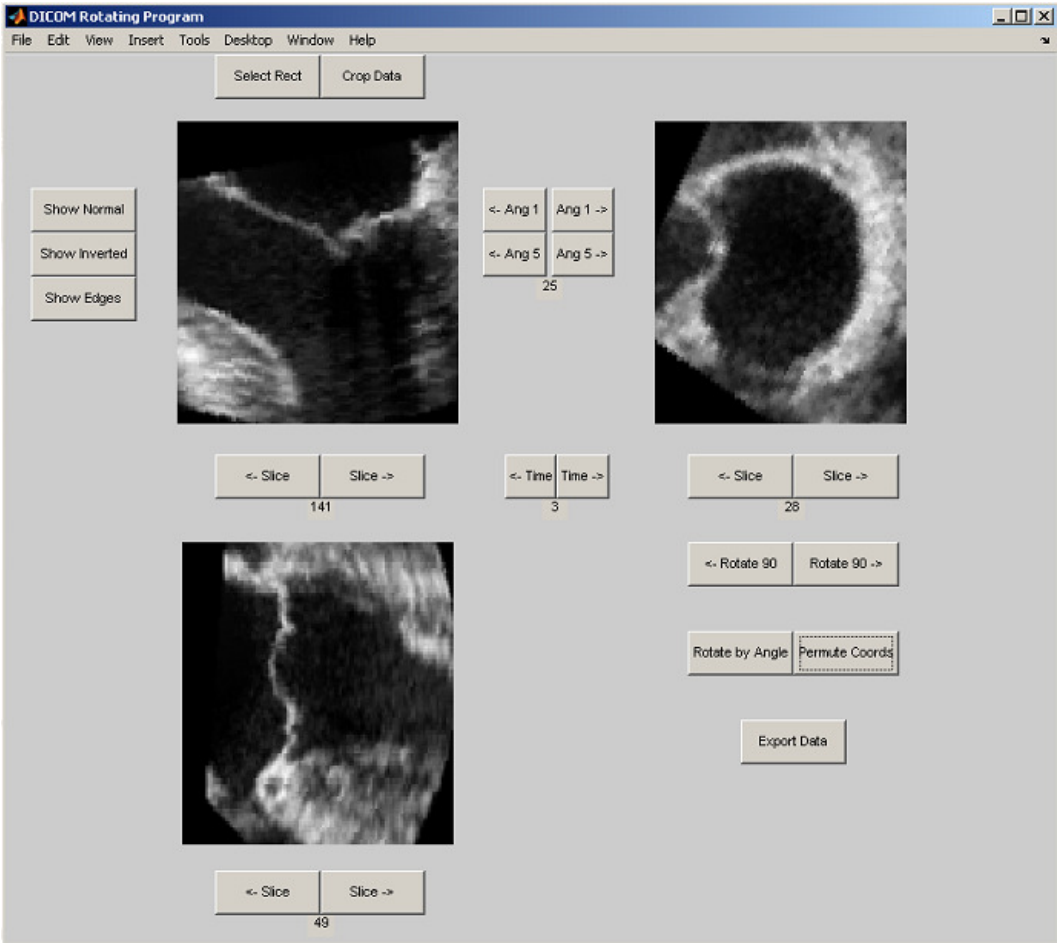
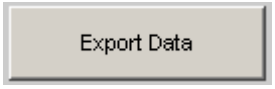


Figure A.3 DICOM rotation program GUI.

15) Once the correct segmentation orientation has been achieve click  to export the Cartesian DICOM to the variable finalDCM.mat. This variable will be saved to your current directory.

The procedure for segmenting the mitral valve from an echocardiography data set for segmentation is as follows:

1) Move the finalDCM.mat file to the *segmentMV_v2* folder.
2) Load the finalDCM.mat file into the workspace. Double-clicking it will accomplish this.

3) Run the *segmentMV* command to start the segmentation program, with the first input being the workspace variable for the DICOM and the second being the spacing between segmentation slices (Figure A.4).

[Ex: *segmentMV*(finalDCM,3)].

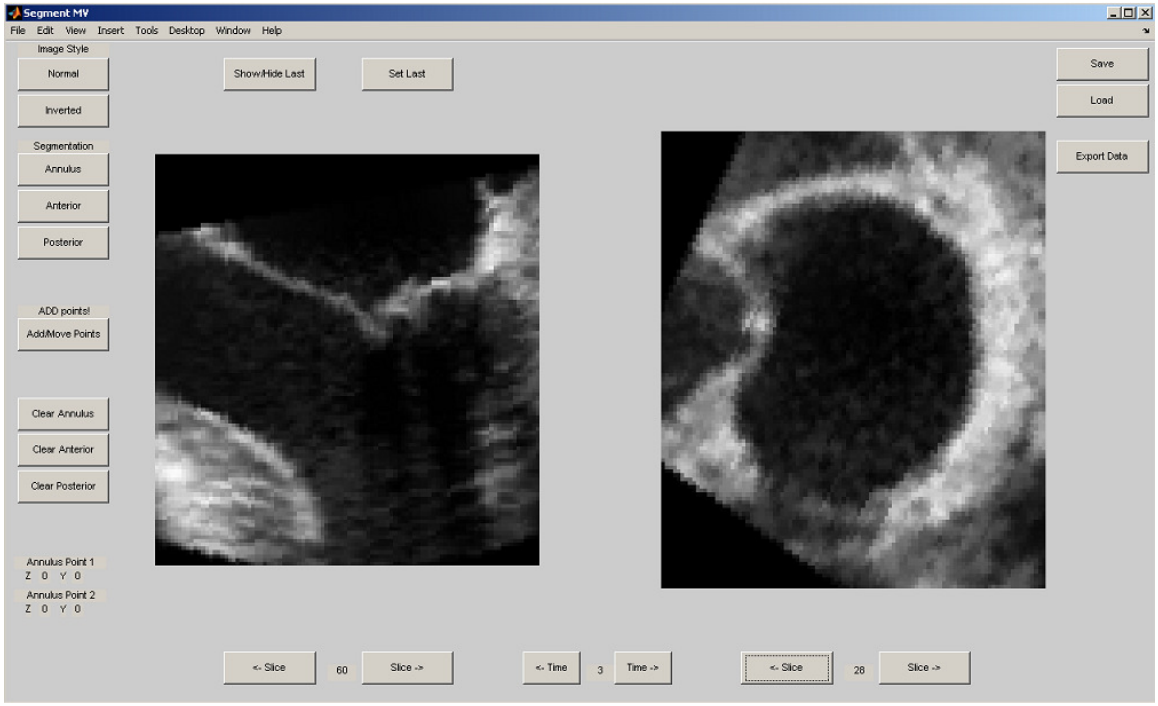


Figure A.4 Segmentation program GUI.

4) Adjust the DICOM time using the time buttons, 

, in the bottom middle of the GUI until the desired time to segment is obtained.

5) Adjust the segmentation images to be shown as normal (Figure A.5) or inverted (Figure A.6) using the image style buttons. This is purely a user preference.

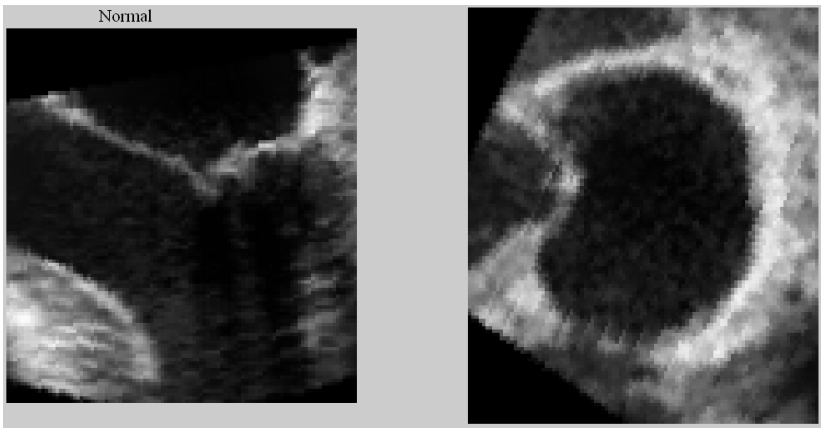


Figure A.5 Example of segmentation views with image style set to normal.

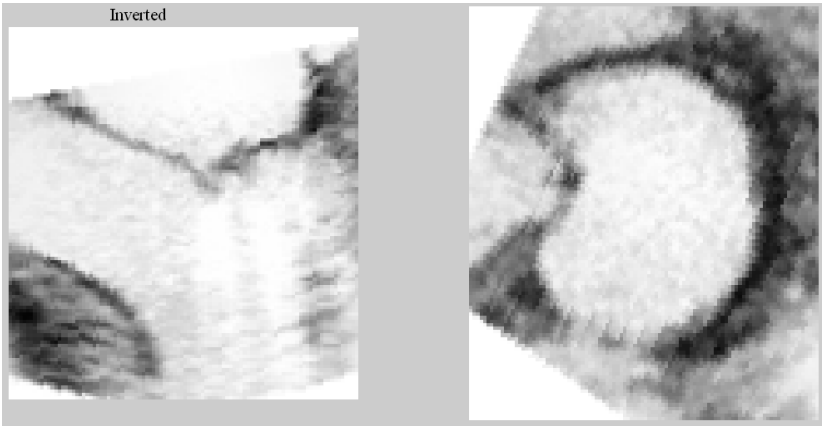


Figure A.6 Example of segmentation views with image style set to inverted.

6) Segment the annulus by clicking 

and selecting the anterior and posterior points of the annulus in the current segmentation slice (left view) (Figure A.7).
    a. The order in which the points are selected does not matter.
    b. The selected annulus points will also appear on the right image.
    c. Clicking 

again will allow the user to move the currently selected point to the newly selected point. Right-click once finished moving the point. The crosshair will disappear.

d. To clear both annulus points click 

.

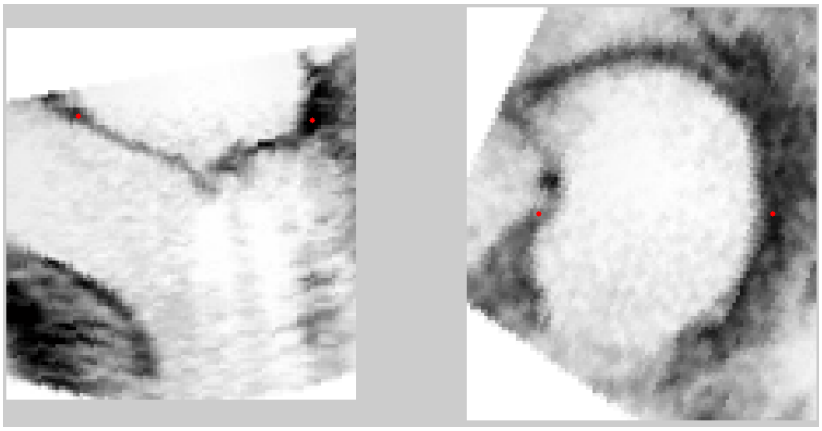


Figure A.7 Example of selected annulus points in the left and right views of the segmentation program. Note the red boths on the annulus in both images.

7) Next either the anterior or posterior leaflet can be segmented by clicking either 

or 

.

   a. Once clicked the segmentation code will begin and crosshairs will appear.
   b. The first point you select will automatically be connected back to the nearest annulus point you selected.
   c. The segmentation curve will be generated in the order you select the points.
   d. The middle of the leaflets should be segmented as changes in gain or other settings will change the thickness of the leaflets.
   e. Once finished segmenting, right-click to end the segmentation loop. The crosshairs will disappear. This confirms the segmentation loop has ended (Figure A.8).

f. Clicking 

or 

again will allow adjustment of the control points in the selected segmentation curve. Once finished, right-click. The crosshair will disappear.

g. If you want to add points to the segmentation curves instead of moving them click 

and the setting will be toggled. The current setting is always displayed above the button.

h. To clear the segmentation for either the anterior or posterior leaflet click 

or 

.

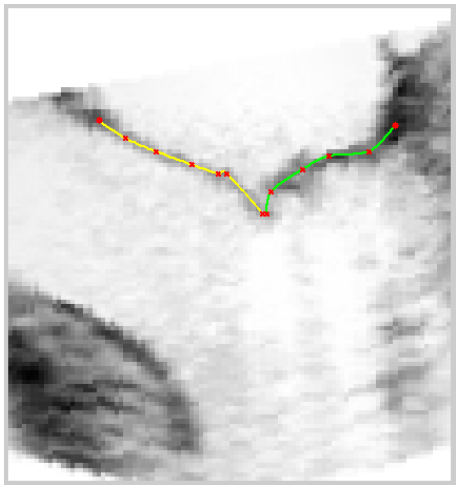


Figure A.8 Example segmentation of the mitral (yellow) and posterior (green) leaflets.

8) Next click one of the slice change buttons, 

, under the left image to change to the image to a neighboring slice.

9) To see the segmentation curves from the last neighboring slice segmented, click 

. This will displayed the last neighboring slice segmented until the button is clicked again. The following is an example of what will be shown over a neighboring segmentation slice (Figure A.9).

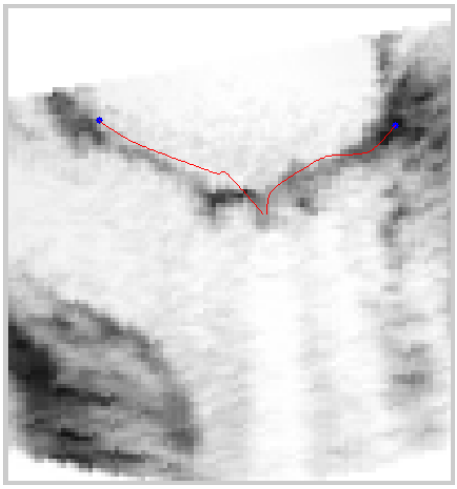


Figure A.9 Example of a segmentation slice with the "Show Last" function enabled.

10) To set the segmentation shown, click 

. Once set, the segmentation can be adjusted by clicking 

, 

, or 

. The following is an example of a neighboring image that has had a neighboring segmentation set as its segmentation. It is clear the segmentation needs to be adjusted.

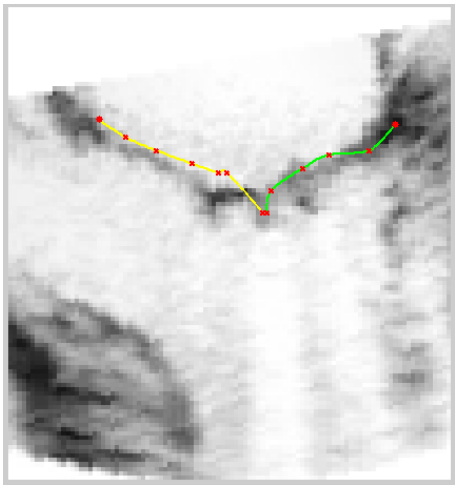
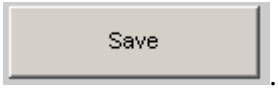


Figure A.10 Example of a segmentation slice after the "Set Last" function was used.

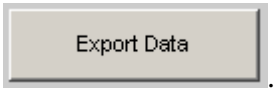
11) To save the current segmentation progress click 

.

12) To load previous segmentation progress click 

.

13) Repeat steps 6-8 until the entire valve is segmented.

14) After the valve is completely segmented, click 

. All segmentation data will be saved and PLY mesh files for the anterior, posterior, and both leaflets will be created. All files are stored in the data folder in the segmentMV directory.

15) If the measurements were taken *in vitro* then they needs to be corrected using the *echoCorrect* function. The following is an example of the MATLAB command required to use this function:

    echoCorrect('filename',[x y z]);

    where *'filename'* is the filename of the PLY file to be corrected, and *[x y z]* is the size of the 3D data set that was segmented. Before this command is run the m-file should be examined to ensure the corrected *in vitro* setup variables are set. These

are located at the beginning of the m-file and are clearly noted. The new PLY file will have the suffix "_scaled".

16) Once the meshes have been exported, they can be refined using either the Loop, Butterfly, or Interslice methods. The following are examples of the MATLAB command required to use each method:

plyLoopRefine('filename');

plyButtRefine('filename);

plyInterslice('filename',n);

where *'filename'* is the filename of the PLY file to be refined in quotes, and *n* is the number of slices to resample the mesh to using the Interslice method. The new PLY file created by each of these will add a suffix to show it has been refined by the method used.

APPENDIX B

SEGMENTATION TOOL CODE GUIDE

This section will discuss the structure and function of the MATLAB codes associated with this study. This includes the codes for the DICOM rotation program, the segmentation program, and the mesh refinement functions.

B.1 Rotation Program

The rotation program consists of multiple MATLAB m-files. Most are linked to the *rotateMV* file. This file sets up all of the required variables, loads the Cartesian DICOM, and displays the GUI. The following is a hierarchy of the codes of the rotation program and a brief description of their function. They are all m-files. If the code is associated with a button in the GUI, the button text is shown in parenthesis.

1) *readDicom3Dedited* – reads the Cartesian DICOM into MATLAB
2) *scaleDicom* – scales DICOM to obtain 0.5 mm voxels
    a. This function takes in a Cartesian DICOM imported into MATLAB. The Cartesian DICOM is a four dimensional matrix. Three dimensions are spatial coordinates and the fourth dimension is time. To scale the three directions the MATLAB function *imresize* was used. This function scales a 2D image to a user defined width and height in pixels. To use this 2D function with the 3D spatial coordinates, two *for* loops were used to traverse each timepoint. First, 2D slices in the *yz* plane were scaled to achieve 0.5 mm spacing. Second, 2D slices in the *xy* plane were scaled to achieve 0.5 mm spacing. This accounts for all three directions.
3) *rotateMV*
    a. *angleDraw* – displays the image in each of the three views

b. *angleInverte*d (Show Inverted) – sets the image type to inverted
c. *angleNorma*l (Show Normal) – sets the image type to normal
d. *angRotate* (Rotate by Angle) – rotates the entire DICOM by the selected angle
e. *cropDCM* (Crop Data) – crops the entire DICOM by the selected rectangle
f. *exportDCM* (Export Data) – exports the final DICOM to finalDCM.mat
g. *nextSliceAngle* (Slice->, top-left) – increases the slice displayed in the top-left view
h. *nextSliceAngleY* (Slice->, top-right) – increases the slice displayed in the top-right view
i. *nextSliceAngleZ* (Slice->, bottom-left) – increases the slice displayed in the bottom-left view
j. *nextTimeAngle* (Time->) – increases the time displayed in all views
k. *prevSliceAngle* (<-Slice, top-left) – decreases the slice displayed in the top-left view
l. *prevSliceAngleY* (<-Slice, top-right) – decreases the slice displayed in the top-right view
m. *prevSliceAngleZ* (<-Slice, bottom-left) – decreases the slices displayed in the bottom-left view
n. *prevTimeAngle* (<-Time) – decreases the time displayed in all views
o. *r90clock* (<-Rotate 90) – sets rotation to clockwise and calls rotateCoord
p. *r90counter* (Rotate 90->) – sets rotation to counter-clockwise and calls rotateCoord
q. *rotateC1* (Ang 1->) – rotates the top-left view by 1 degree clockwise
r. *rotateC5* (Ang 5->) – rotates the top-left view by 5 degrees clockwise
s. *rotateCC1* (<-Ang 1) – rotates the top-left view by 1 degree counter-clockwise

t. *rotateCC5* (<-Ang 5) – rotates the top-left view by 5 degrees counter-clockwise

u. *rotateCoord* – rotate DICOM by 90 degrees in a clockwise or counterclockwise direction

v. *selectRect* – allows the user to define a cropping rectangle by two points in the top-left view

w. *switchCoord* – permutes the (x, y, z) coordinates of the DICOM to (y, z, x) using the MATLAB function *permute*

The functions within *rotateMV* that manipulate the Cartesian DICOM use a *for* loop to push through the 3D stack and manipulate each *yz* plane to create a new 3D volume. The following is a list of these functions and the 2D MATLAB function they use to manipulate the mesh.

1) *angRotate* – uses the MATLAB function *imrotate*
2) *cropDCM* – uses the MATLAB function *imcrop*
3) *rotateCoord* – uses the MATLAB function *rot90*

B.2 Segmentation Program

The segmentation program consists of multiple MATLAB m-files. All are linked to the *segmentMV* file. This file sets up all of the required variables, loads the Cartesian DICOM, and displays the GUI. The following is a hierarchy of the codes of the segmentation program and a brief description of their function. They are all m-files. If the code is associated with a button in the GUI, the button text is shown in parenthesis.

1) *rotateMV*

a. *clearAnnulus* (Clear Annulus) – clears the annulus points from the current segmentation slice

b. *clearAnt* (Clear Anterior) – clears the anterior control points from the current segmentation slice
c. *clearPost* (Clear Posterior) – clears the posterior control points from the current segmentation slice
d. *exportData* (Export Data) – samples the segmentation curves, meshes the leaflets, sorts the annulus points, and then saves all of the segmentation data
    i. *meshLeaflets* – generates meshes from the sampled segmentation curves and saves the PLY files for the anterior leaflet, posterior leaflet, and both leaflets as a single mesh
    ii. *resampleSeg* – samples the segmentation curves from each slice at the same number
    iii. *sortAnnulus* – sorts the annulus so that the points are ordered around the annulus and saves the data
e. *loadSeg* (Load) – loads a previously saved segmentation
f. *nextSlice* (Slice->, left) – increases the current slice number of the left view
g. *nextSliceY* (Slice->, right) – increases the current slice number of the right view
h. *nextTime* (Time->) – increases the time displayed for both views
i. *prevSlice* (<-Slice, left) – decreases the current slice number of the left view
j. *prevSliceY* (<-Slice, right) – decreases the current slice number of the right view
k. *prevTime* (<-Time) – decreases the time displayed by both views
l. *redrawData* – displays any available segmentation data for the left view
m. *redrawDataY* – displays any available segmentation data for the right view

n. *saveSeg* (Save) – saves the current segmentation progress to the data folder

o. *selectAnnulus* (Annulus) – allows the user to select two annulus points in the left view

p. *selectAnterior* (Anterior) – allows the user to generate a curve for the anterior leaflet by selecting the control points of a J_0-spline

q. *selectPosterior* (Posterior) - allows the user to generate a curve for the posterior leaflet by selecting the control points of a J_0-spline

r. *setLast* (Set Last) – sets the segmentation data of to the current slice to that from the last visited neighboring slice

s. *showInverted* (Inverted) – sets the image type to be displayed as inverted

t. *showLast* (Show/Hide Last) – toggles if the data from the last neighboring slice should be displayed or not

u. *showNormal* (Normal) – sets the image type to be displayed as normal

v. *toggleAddMove* (Add/Move Points) – toggles if points are added or moved when adjusting the leaflet control points

w. *tweak4pt* – code to generate a J_0-spline from a set of control points

    i. *applyLaplace* – applies the calculated movement to the control polygon

    ii. *computLaplace* – calculates the movement of the control polygon

    iii. *refinePts* – refines a control polygon by inserting new points at the midpoints of the current points

    iv. *sn* – function to move to the next point in the control polygon

    v. *sp* – function to move to the previous point in the control polygon

x. *writePLY* – function that writes a PLY file given a set of vertices, connectivity, and a filename

B.3 Mesh Refinement Functions

All of the codes for the Loop, Butterfly, and Interslice mesh refinement schemes are in MATLAB m-files. The following is a hierarchy of the codes for each of these refinement schemes and a brief description of their function.

1) *plyLoopRefine* – refines a given PLY filename according to the Loop refinement method
2) *plyButtRefine* – refines a given PLY filename according to the Butterfly refinement method
3) *plyInterslice* – refines a given PLY filename according to the Interslice refinement method
    a. *tweak4pt* – code to generate a J_0-spline from a set of control points
        i. *applyLaplace* – applies the calculated movement to the control polygon
        ii. *computLaplace* – calculates the movement of the control polygon
        iii. *refinePts* – refines a control polygon by inserting new points at the midpoints of the current points
        iv. *sn* – function to move to the next point in the control polygon
        v. *sp* – function to move to the previous point in the control polygon

REFERENCES

1. *Fact Sheet: The Top Ten Causes of Death*, 2008, World Health Organization.
2. Lloyd, J., *Heart Disease and Stroke Statistics-2010 Update: A Report From the American Heart Association (vol 121, pg e46, 2010).* Circulation, 2010. **121**(12): p. E260-E260.
3. Padala, S.M., *Mechanics of the Mitral Valve after Surgical Repair: An In-Vitro Study*, in *Bioengineering*2010, Georgia Institute of Technology: Atlanta, GA.
4. Levine, R.A., et al., *3-Dimensional Echocardiographic Reconstruction of the Mitral-Valve, with Implications for the Diagnosis of Mitral-Valve Prolapse.* Circulation, 1989. **80**(3): p. 589-598.
5. Lam, J.H.C., Ranganat.N, and E.D. Wigle, *Morphology of Human Mitral Valve .1. Chordae Tendineae - A New Classification.* Circulation, 1970. **41**(3): p. 449-&.
6. Beique, F., D. Joffe, and S. Kleiman, *An introduction to transoesophageal echocardiography .1. Basic principles.* Canadian Journal of Anaesthesia-Journal Canadien D Anesthesie, 1996. **43**(3): p. 252-277.
7. Støylen, A. *Basic ultrasound, echocardiography and Doppler for clinicians*. 2010; Available from: http://folk.ntnu.no/stoylen/strainrate/Ultrasound/.
8. Stern, B. *The Basic Concepts of Diagnostic Ultrasound*. Medical Imaging 1987; Available from: http://www.yale.edu/ynhti/curriculum/units/1983/7/83.07.05.x.html.
9. Daly, *Piezoelectric Transducers*, 2009, University of Washington.
10. Shung, *The Princeple of Multidimensional Arrays.* European Journal of Echocardiography, 2002(3): p. 149-153.
11. Garcia-Orta, R., et al., *Three-dimensional versus two-dimensional transesophageal echocardiography in mitral valve repair.* Journal of the American Society of Echocardiography, 2007. **20**(1): p. 4-12.
12. Daimon, M., et al., *Dynamic change in mitral annular area and motion during percutaneous mitral annuloplasty for ischemic mitral regurgitation: Preliminary animal study with real-time 3-dimensional echocardiography.* Journal of the American Society of Echocardiography, 2007. **20**(4): p. 381-388.
13. Little, S.H., et al., *Dynamic Annular Geometry and Function in Patients with Mitral Regurgitation: Insight From Three-Dimensional Annular Tracking.* Journal of the American Society of Echocardiography, 2010. **23**(8): p. 872-879.
14. Veronesi, F., et al., *Quantification of mitral apparatus dynamics in functional and ischemic mitral regurgitation using real-time 3-dimensional echocardiography.* Journal of the American Society of Echocardiography, 2008. **21**(4): p. 347-354.
15. Tsukiji, M., et al., *Three-dimensional quantitation of mitral valve coaptation by a novel software system with transthoracic real-time three-dimensional echocardiography.* Journal of the American Society of Echocardiography, 2008. **21**(1): p. 43-46.
16. Silvestry, F.E., et al., *Echocardiography Guided Interventions (vol 22, pg 213, 2009).* Journal of the American Society of Echocardiography, 2009. **22**(4): p. 336-336.
17. Biaggi, P., M. Greutmann, and A. Crean, *Utility of Three-Dimensional Transesophageal Echocardiography: Anatomy, Mechanism, and Severity of*

Regurgitation in a Patient with an Isolated Cleft Posterior Mitral Valve. Journal of the American Society of Echocardiography, 2010. **23**(10).
18. Maffessanti, F., et al., *Quantitative Analysis of Mitral Valve Apparatus in Mitral Valve Prolapse Before and After Annuloplasty: A Three-Dimensional Intraoperative Transesophageal Study.* Journal of the American Society of Echocardiography, 2011. **24**(4): p. 405-413.
19. Mikic, I., S. Krucinski, and J.D. Thomas, *Segmentation and tracking in echocardiographic sequences: Active contours guided by optical flow estimates.* Ieee Transactions on Medical Imaging, 1998. **17**(2): p. 274-284.
20. Bashein, G., M.E. Legget, and P.R. Detmer, *Pointwise assessment of three-dimensional computer reconstruction of mitral leaflet surfaces from rotationally scanned echocardiograms in vitro.* Journal of the American Society of Echocardiography, 2004. **17**(3): p. 239-246.
21. Martin, S., *Fast Segmentation of the Mitral Valve Leaflet in Echocardiography* in *Computer Vision Approaches to Medical Image Analysis*, M.S. Reinhard R. Beichel, Editor. 2006, Springer-Verlag: Berlin, Germany. p. 225-235.
22. Shang, Y.F., et al., *Region competition based active contour for medical object extraction.* Computerized Medical Imaging and Graphics, 2008. **32**(2): p. 109-117.
23. Ryan, L.P., et al., *A methodology for assessing human mitral leaflet curvature using real-time 3-dimensional echocardiography.* Journal of Thoracic and Cardiovascular Surgery, 2008. **136**(3): p. 726-734.
24. Schneider, R.J., et al. *Proceedings of the Sixth IEEE international conference on Mitral annulus segmentation from three-dimensional ultrasound*. in *Symposium on Biomedical Imaging: From Nano to Macro*. 2009. Boston, Massachusetts, USA
25. Macnab, A., Jenkins, N., Ray, S, *Three-dimensional echocardiography is superior to multiplane transoesophageal echo in the assessment of regurgitant mitral valve morphology.* European Journal of Echocardiography, 2003.
26. Valocik, G., Kamp, O., Visser, C., *Three-dimensional echocardiography in mitral valve disease.* European Journal of Echocardiography, 2005.
27. Adams, D., Anyanwu, A., Sugeng, L., Lang, R., *Degenerative Mitral Valve Regurgitation: Surgical Echocardiography.* Current Cardiology Reports, 2008.
28. Salcedo, E., Quaife, R., Seres, T., Carroll, J., *A Framework for Systematic Characterization of the Mitral Valve by Real-Time Three-Dimensional Transesophageal Echocardiography.* Journal of the American Society of Echocardiography, 2009.
29. Turk, G., *The PLY Polygon File Format*, 1994, Stanford University: Stanford, CA.
30. Rossignac, J., *Curves*, 2009, Georgia Institute of Technology: Atlanta, GA.
31. Rossignac, J. and S. Schaefer, *J-splines.* Computer-Aided Design, 2008. **40**(10-11): p. 1024-1032.
32. Rossignac, J., *Education-driven research in CAD.* Computer-Aided Design, 2004. **36**(14): p. 1461-1469.
33. Rossignac, J. and A. Venkatesh, *Ringing: Frugal Subdivision of Curves and Surfaces.* IEEE Computer Graphics and Applications, 2010. **30**(2): p. 22-33.

34. Rossignac, J., Safonova, A., Szymczak, A. *3D compression made simple: Edgebreaker on a Corner Table*. in *Shape Modeling International Conference*. 2001. Genoa, Italy.
35. Loop, C., *Smooth subdivision surfaces based on triangles*, in *Mathematics*1987, University of Utah: Salt Lake City, Utah.
36. Dyn, N., D. Levin, and J.A. Gregory, *A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control.* Acm Transactions on Graphics, 1990. **9**(2): p. 160-169.
37. He, S.Q., et al., *Mitral valve compensation for annular dilatation: In vitro study into the mechanisms of functional mitral regurgitation with an adjustable annulus model.* Journal of Heart Valve Disease, 1999. **8**(3): p. 294-302.
38. Jensen, M.O.J., *Stentless mitral valve fixation: impact on hemodynamic performance*, in *Biomedical Engineering*2000, Georgia Institute of Technology: Atlanta, GA.
39. Jensen, M.O.J., A.A. Fontaine, and A.P. Yoganathan, *Improved in vitro quantification of the force exerted by the papillary muscle on the left ventricular wall: Three-dimensional force vector measurement system.* Annals of Biomedical Engineering, 2001. **29**(5): p. 406-413.
40. Kwon, Y.-H. *DLT Method*. 1998; Available from: http://www.kwon3d.com/theory/dlt/dlt.html.
41. Weisstein, E.W. *Snell's Law*. Available from: http://scienceworld.wolfram.com/physics/SnellsLaw.html.
42. Weisstein, E.W. *Index of Refraction*. From MathWorld--A Wolfram Web Resource.; Available from: http://scienceworld.wolfram.com/physics/IndexofRefraction.html.
43. *Parker Labs Ultrasound Products*. Available from: http://www.parkerlabs.com/ultrasound_products.html.
44. Ltd, B., *Speed of Sound through Materials.*
45. Ophir, J. and T. Lin, *A Calibration-Free Method for Measurement of Sound Speed in Biological Tissue Samples.* Ieee Transactions on Ultrasonics Ferroelectrics and Frequency Control, 1988. **35**(5): p. 573-577.
46. Weisstein, E.W. *Point-Line Distance--3-Dimensional*. Available from: http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html.
47. *Algorithm Tutorials*. Available from: http://www.topcoder.com/tc?d1=tutorials&d2=geometry1&module=Static#line_point_distance.
48. Ericson, C., *Real-Time Collision Dectection*. The Morgan Kaufmann Series in Interactive 3-D Technology, ed. D. Eberly. 2005, San Francisco, CA: Morgan Kaufmann.
49. Weisstein, E.W. *Triangle Area*. Available from: http://mathworld.wolfram.com/TriangleArea.html.
50. Kunzelman, K.S. and R.P. Cochran, *Finite element analysis of the mitral valve.* Journal of Heart Valve Disease, 1993. **2**(3): p. 326-40.
51. Salgo, I.S., et al., *Effect of annular shape on leaflet curvature in reducing mitral leaflet stress.* Circulation, 2002. **106**(6): p. 711-717.

52. Einstein, D.R., et al., *Non-linear fluid-coupled computational model of the mitral valve.* Journal of Heart Valve Disease, 2005. **14**(3): p. 376-385.
53. Kunzelman, K.S., D.R. Einstein, and R.P. Cochran, *Fluid-structure interaction models of the mitral valve: function in normal and pathological states.* Philosophical Transactions of the Royal Society B-Biological Sciences, 2007. **362**(1484): p. 1393-1406.
54. Votta, E., et al., *3-D computational analysis of the stress distribution on the leaflets after edge-to-edge repair of mitral regurgitation.* Journal of Heart Valve Disease, 2002. **11**(6): p. 810-822.
55. Messas, E., et al., *Chordal cutting - A new therapeutic approach for ischemic mitral regurgitation.* Circulation, 2001. **104**(16): p. 1958-1963.